BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/60

A2

(11) International Publication Number: WO 00/54204

(43) International Publication Date: 14 September 2000 (14.09.00)

(21) International Application Number: PCT/US00/06535

(22) International Filing Date: 10 March 2000 (10.03.00)

(30) Priority Data:
09/266,246 10 March 1999 (10.03.99) US

(71) Applicant *(for all designated States except US)*: LIQUIDMARKET, INC. [US/US]; Suite 100, 1983 West 190th Street, Torrance, CA 90504 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: GROULT, Gauthier, H. [FR/US]; Apt. 3, 144 South Roxbury Drive, Beverly Hills, CA 90212 (US). ROUAIX, Francois [FR/US]; Apt. 307, 1310 Esplanade, Redondo Beach, CA 90277 (US). TREVITHICK, Matthew, D. [CA/US]; 100 Driftwood Street, #15, Marina del Rey, CA 90292 (US).

(74) Agents: ZIMMER, Kevin, J.; Cooley Godward LLP, 3000 El Camino Real, Five Palo Alto Square, Palo Alto, CA 94306–2155 (US) et al.

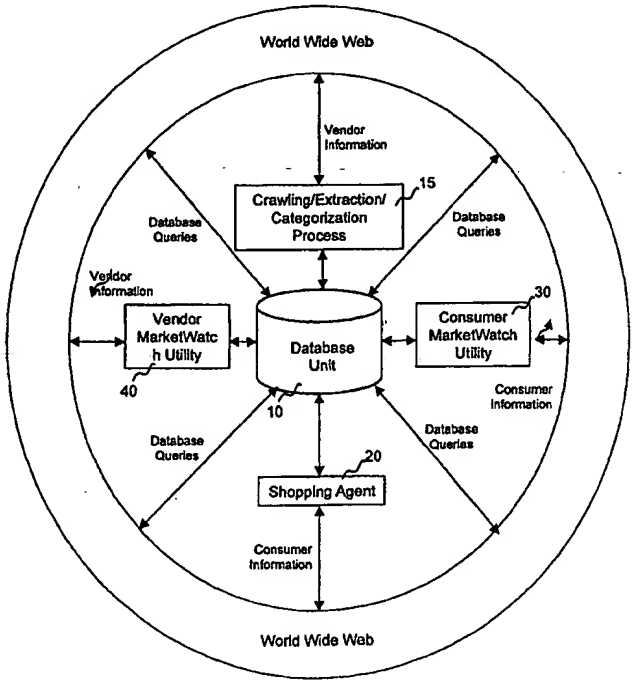
(81) Designated States: CA, IL, JP, SG, US, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: INTERNET–BASED EXCHANGE FOR PRODUCTS AND SERVICES

(57) Abstract

A computer–based system and method for facilitating transactions between vendors and buyers is disclosed herein. A database containing information relating to a plurality of product or service offerings is compiled, and is preferably implemented in an object–oriented format. The database is structured so as to reflect relevant aspects of a market for the product and service offerings. Bids identifying one of the offerings within the database, and an associated offer price, may be entered into a computer. Upon entry of a bid, a buyer identifier associated with the bid is optionally generated. The bid is then made available to one or more vendors of the offering; and a buyer corresponding to the buyer identifier is notified upon any vendor acceptances of the bid. In one implementation, the database is updated based upon the results of an internet–based search for product information extracted from the internet sites of various vendors. The vendor–supplied information is assigned to various product categories, and is stored within the database on the basis of these categories. The database may be searched in response to buyer queries for particular products or services. The items of information relating to product offerings or descriptions collected in response to a buyer query are preferably ranked in accordance with criteria determinative of the relevancy of each such item. This allows information obtained from potentially thousands of web sites to be categorized within a single database, and facilitates performance of extremely fast searches in response to buyer queries for product information.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

INTERNET-BASED EXCHANGE FOR PRODUCTS AND SERVICES

FIELD OF THE INVENTION

The present invention relates generally to electronic commerce and more particularly to a system and method in which creation of an internet-based exchanged for products and services is facilitated through automatic aggregation of product information from multiple vendors.

BACKGROUND OF THE INVENTION

Considerable effort has recently been directed to development of systems and methods for conducting commerce via an electronic means. For example, in an effort to overcome the limitations and disadvantages of physical auctions, systems have been developed to hold electronic auctions over the Internet using electronic mail ("E-mail"). In E-mail auctions, an auction catalog is electronically mailed to people interested bidding. Subsequently, bidders submit their bids on individual lots to an auctioneer via e-mail. The auctioneer reads the electronic mail bids and enters them in a database of bids. When the auction closes, the auctioneer notifies the winning bidders, usually via e-mail, and ships the merchandise to the winning bidders.

There are several disadvantages to e-mail auctions. First, a human auctioneer is required to prepare the auction catalog and to read and process the electronic mail bids. This takes a considerable amount of effort in a large auction. Secondly, it is difficult to keep the bidders updated as to the current high bids on the various items. Electronic mail on most large public networks, such as the Internet, is lower priority traffic than most, meaning it can take several hours for bids to reach the auctioneer and for bidding updates to reach the bidders. Thirdly, as the auction closing draws near, the volume of bids may prohibit the auctioneer from sending out high bid information to the bidders because of the time involved in reading the electronic mail bids and in entering them into the bid database.

A recent innovation applied to e-mail auctions is the use of the Internet's World Wide Web (the "Web") facility to post descriptions of the merchandise and show the current high bids. This innovation provides the advantage of eliminating the need to electronically mail bidding updates to bidders. And since Web traffic is much higher priority on the Internet, bidders suffer less of a time lag in seeing updated Web pages. However, a human auctioneer in still involved and is required to manually

process the electronic mail bids, enter them into the bid database, and to update the Web pages with current high bid information.

Sales firms other than auction houses have also used the Internet's World Wide Web facility to post descriptions of their merchandise and to offer the merchandise for sale at a set price. These systems are automated and are capable of accepting an order from a customer by having that customer fill out an online order form. This order information is taken by the system and placed into an order database or accounting system which then processes the order. However, such systems sell merchandise only at a fixed price and do not allow merchandise to be auctioned off, or to have their prices dynamically adjusted in an interactive manner in response to bids and other market conditions such as supply and demand. These systems are "seller driven" in the sense that the seller sets a price and the buyer decides whether or not to accept that price. Moreover, distinct user interfaces for each vendor require repetitive entry of user information to complete purchases. Users are burdened with maintaining accounts and passwords with each Internet vendor with whom they do business.

A "buyer-driven" system may be characterized as one in which buyers specify the parameters of a desired transaction to a group of sellers. An example of a regularly used buyer-driven process is the technique used by large commercial or governmental enterprises desiring to purchase significant quantities of goods or services at the lowest possible price. To begin, the buying entity composes a written specification setting for the quantities and requirements or the products desired to be purchased. This document is generally called a "Request for Proposal", or "RFP". Once finalized, RFPs are distributed to a list of known potential suppliers.

Potential suppliers which identify an RFP that they might be able to fulfill, will first evaluate it to decide whether or not to invest the necessary time and effort to submit a formal proposal. Typically, some number of suppliers submit binding proposals to the buyer by a deadline established in the RFP. Once submitted, proposals are then evaluated by the buyer. One proposal is usually selected and the corresponding supplier notified that it has "won" the business at the price quoted.

Individual consumers cannot effectively participate in such buyer-driven systems because they generally do not have the buying power and resources of large organizations. Some customers have found ways to group together in order to achieve some measure of the volume buying power enjoyed by large organizations. Many

consumers, however, are deterred from joining buying groups because of the groups' various requirements and limitations.

In an effort to enable ordinary consumers to participate in "buyer-driven" electronic commerce, at least one system has recently been proposed in order to afford consumers the opportunity to communicate a binding purchase offer to a group of potential sellers. The system is designed to allow sellers to bind a buyer to a contract based upon the buyer's purchase offer. In the proposed system a central server receives binding purchase offers from prospective buyers. The central server makes purchase offers available globally to potential sellers. Potential sellers then have the option to accept a purchaser offer and thus bind the corresponding buyer to a contract. The seller may also issue a binding counteroffer in response to a purchase offer, which is forwarded to the buyer by way of the central server. The buyer is then given the option of accepting the counteroffer and thereby binding the seller to a contract.

This type of "reverse auction" system improves upon prior auction arrangements by generally dispensing with the need for a human auctioneer. Although this may enhance efficiency, implementation of other aspects of such systems may require significant technological overhead. For example, since existing reverse auction systems purport to establish a binding contractual relationship between buyer and seller, it is necessary to establish the authenticity of purchase orders and of corresponding acceptances. To the extent that disputes arise with regard to offer, acceptance or other aspects of contract formation (whether relating to authenticity or otherwise), mechanism will preferably be provided for resolving such disputes. Complexity is further increased by the need for configuring the system to handle most or all of the payment options offered by participating sellers (e.g., credit cards, personal checks, electronic funds transfer, digital money, etc.).

Existing electronic reverse auction systems are also not known to provide buyers with the type of comprehensive pricing information from multiple vendors which would facilitate the placement of well-informed bids. For example, a buyer seeking significant savings may desire to place a bid which is at least 50% below the lowest published price for a given product. In cases where a product is needed relatively soon, buyers may seek only a relatively modest discount (e.g., 10%) off of the lowest generally available price in order to increase the likelihood that the bid price will be accepted within a reasonable time. Unfortunately, in order to obtain such pricing information buyers must conduct their own research either online or

through more conventional means. Since existing online search engines often yield reams of unhelpful information in response to even relatively targeted queries, performing independent online product pricing and feature research can frequently be a time consuming, unproductive process. Even when pertinent information is retrieved, it can be in relatively "raw" form and difficult to interpret.

Ideally, it would be desirable if product and service information from multiple vendors could be collected within a single database in order to enable users to efficiently find information of interest. Unfortunately, such vendor information is not structured in a standardized manner, which complicates efforts to collect it in a centralized repository. Given the improbability of structural standardization of vendor sites throughout the Web, any effort at establishing a centralized database of vendor product information will likely require the development of techniques for extracting information from vendor sites which is formatted in a nonstandard manner.

Existing reverse auction systems are also not known to enable buyers to collectively increase their leverage relative to sellers by facilitating the aggregation of bids for a given product. To the extent a sufficiently large number of bids could be aggregated, a vendor would potentially be inclined to sell below its published price in order to achieve an increased sales volume.

It is thus apparent that existing forms of electronic commerce, such as current reverse auction systems, have not as yet enabled creation of an efficient exchange for goods and services analogous to the financial markets. In this regard the existing price disparity among vendors of similar or identical products evidences an inefficiency in the dissemination of relevant price and product information to interested buyers. In contrast, the fact that a given security or commodity trades at approximately the same currency-adjusted price on different exchanges is attributable to the efficiency with which information is communicated among participating traders.

It would thus be desirable to develop a system for electronic commerce enabling goods and services to be exchanged with same degree of efficiency currently accompanying the exchange of securities, commodities and other instruments by way of the financial markets. Such a system would preferably enable a wide array of products and services to be purchased essentially as commodities, with minimal price disparity existing among vendors of identical offerings.

SUMMARY OF THE INVENTION

The present invention provides a computer-based system and method for facilitating transactions between vendors and buyers. In accordance with the invention, a database containing information relating to a plurality of product or service offerings is compiled. In a preferred embodiment the database is implemented in an object-oriented format and structured so as to reflect relevant aspects of the market for the product and service offerings. Bids identifying one of the offerings within the database, and an associated offer price, may be entered into a computer. Upon entry of a bid, a buyer identifier associated with the bid is optionally generated. The bid is then made available to one or more vendors of the offering; and a buyer corresponding to the buyer identifier is notified upon any vendor acceptances of the bid.

In one embodiment, the database is updated based upon the results of an internet-based search for product information extracted from the internet sites of various vendors. The vendor-supplied information is assigned to various product categories, and is stored within the database on the basis of these categories. The database may be searched in response to buyer queries for particular products or services. The items of information relating to product offerings or descriptions collected in response to a buyer query are preferably ranked in accordance with criteria determinative of the relevancy of each such item.

The present invention advantageously allows information obtained from potentially thousands of web sites to be collected and categorized within a single database, preferably implemented in an object-oriented format. This unique organization of product information within the database facilitates performance of extremely fast searches in response to buyer queries for product information. In addition, the database is designed to easily accommodate updates of vendor from surveyed web sites.

BRIEF DESCRIPTION OF THE DRAWINGS

In the accompanying drawings:

FIG. 1 provides a generalized representation of system architecture of the present invention.

FIG. 2 is a block diagram providing an overview of principal components within the electronic commerce system of the present invention.

FIG. 3 provides a more detailed representation of a database server unit and central database of the present invention.

FIG. 4 is a block diagram depicting the logical relationship among the software objects comprising a preferred implementation of an object-oriented market model included within the central database.

FIG. 5 is a logical diagram representative of a preferred application architecture of the central database.

FIG. 6 provides an illustrative representation of the architecture of a crawler component operative in conjunction with the database server.

FIG. 7 illustratively represents the process by which the indexer and extractor operate upon HTML documents (i.e., Web pages) stored within the crawler page repository.

FIGS. 8 and 9 illustratively represent the extraction of relevant attribute-value pairs from an exemplary HTML document in accordance with the present invention.

FIG. 10 illustratively represents an exemplary sequence of processes performed by a categorizer component within the database server.

FIG. 11 is a flow chart representative of the processing performed by the shopping agent of the present invention during interaction with a potential purchaser.

FIG. 12 is a flow chart depicting an exemplary manner in which bids are placed by users, and evaluated by vendors, in accordance with the present invention.

FIG. 13 is a flow chart depicting an exemplary sequence of steps involved in implementing a market watch utility of the present invention.

FIG. 14 provides a flow chart representation of the processes occurring during a bid matching process of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

GLOSSARY

The following commonly accepted acronyms and abbreviations are among those utilized in the present specification:

CGI Common Gateway Interface. A specification for interfacing content-providing applications to Web servers.

ASP Active Server Page (Microsoft "scripting" language for dynamic construction of Web pages)

SGML Standard Generalized Markup Language

DTD Document Type Definition, the analog of a grammar for SGML

URL Uniform Resource Locator, the abstract identifier used to locate resources on the Internet.

AST Abstract Syntax Tree

SYSTEM OVERVIEW

FIG. 1 provides a generalized representation of system architecture of the present invention. Referring to FIG. 1, the present invention facilitates online shopping by providing a comprehensive and current database 10 of products available for purchase via the Internet. The database 10 is created using crawling, extraction and categorization processes 15 disposed to operate upon information available from vendor web sites on the Internet. The database 10 is utilized when consumers invoke a shopping agent 20 or consumer market watch utility 30, and when vendors invoke a vendor market watch utility 40. In a preferred embodiment a product search engine of the shopping agent 20 affords users the opportunity to intuitively locate, preferably through use of hierarchical menus and keyword searches, one or more suitable vendors for specific products. The shopping agent 20 simplifies the purchase process by capturing user information and automatically completing forms required to process transactions. The consumer market watch utility 30 enables users to submit bids specifying criteria required to be met by vendors desiring to sell an identified product. Similarly, the vendor market watch utility 40 allows participating vendors to observe the content of bids submitted by consumers for particular products. The user is notified when a vendor has agreed to meet the specified criteria, thereby permitting consummation of a commercial transaction acceptable to both vendor and consumer.

Referring to FIG. 2, the present invention preferably includes a Web server 100 connected to a plurality of vendor interfaces 110 and to a plurality of buyer

interfaces 120. The vendor interfaces 110 and buyer interfaces 120 are linked to the Web server 100 through a conventional Internet connection via modems 122 and 124, respectively. The Web server 100 functions to both receive bids from buyers and to transmit bid information in response to queries by sellers. The Web server 100 is also linked, via a Web page service 130, to a connected to a database server 150. The database server 150 includes software applications disposed to extract and categorize relevant product information from vendor web sites, and to regularly updates a central database 160 using the resultant information. As is discussed below, the central database 160 preferably comprises an object-oriented database designed to facilitate the form of electronic commerce contemplated by the present invention.

The apparatus of FIG. 2 enables buyers to browse the central database 160 in order to ascertain the lowest available price for a selected product. Interested buyers may then post bids, which are made available to potential vendors. Upon notifying a buyer that its bid has been accepted by a vendor, the buyer is charged a brokerage fee and is given the opportunity to enter into a commercial transaction with the accepting vendor. In a preferred embodiment the brokerage fee is based upon a percentage of the difference between the lowest available product price and the buyer's bid, and accrues irrespective of whether the buyer ultimately consummates a commercial transaction with the accepting vendor.

Turning now to FIG. 3, an exemplary physical realization of the database server 150 is seen to include a central processing unit ("CPU") 172, random access memory ("RAM") 174, read only memory ("ROM") 178, and a payment processor module 184. The realization of the database server 150 depicted in FIG. 3 preferably includes sufficient memory and processing capability to handle a high volume of transactions. RAM 174 and ROM 178 collectively contain a variety of data structures and information, including an operating system, program objects, and user data. Those skilled in the art will realize that hard disk drives, optical disk drives, and other storage devices containing logically segmented storage locations may also be used for these purposes.

The payment processor 184 may be realized using one or more conventional microprocessors disposed to support the transfer of brokerage fees and other charges accruing in connection with buyers' usage of the system of the present invention. Payment processor 184 may also be configured as part of CPU 172. Processing of credit card transactions by payment processor 184 may be supported with

commercially available software, such as the Secure Webserver manufactured by Open Market, Inc. This server software transmits credit card numbers electronically over the Internet to servers located at the Open Market headquarters where card verification and processing is handled.

Referring to FIG. 3, the central database 160 is realized using a direct access storage device, such as magnetic disk storage, in which object-oriented data files are stored. The central database contains an object-oriented market model 210 used in effecting the transaction brokerage contemplated by the present invention. As is described in detail below with reference to FIG. 4, the market model 210 contains objects relating to buyers, vendors, product specifications and attributes, product prices, and product brands. In accordance with the present invention, information pertinent to most of the objects within the market model 210 is regularly extracted from vendor web sites in a largely automated manner via the database server 150. In addition, certain information within the market model 210 pertinent to buyers is collected on the basis of buyer interaction with the buyer interface 120, and includes buyer profiles, parameters relating to the market watch utility, and shopping history.

The central database also includes a shopping agent basket 212 for storing information utilized by a shopping agent application (described below) operative on the database server 150. The shopping agent basket 212 contains a user information object 214, and a vendor form object 216.

As is indicated by FIG. 3, the Web page service 130 is seen to include an inbound service component 130a and an outbound service component 130b. The inbound service component 130a is responsible for collecting form-based information such as queries or profiles from users and vendors. Similarly, the outbound service component 130b converts data from the database server 150 into HTML content delivered to vendors and buyers via the Web server 100.

PROCESS OVERVIEW

Referring again to FIG. 2, in a preferred embodiment of the invention a crawler 250, indexer 254, extractor 258 and categorizer 262 are collectively disposed to capture, organize, and categorize consumer and vendor information pertaining to electronic commerce. Specifically, the crawler 250 retrieves pages (i.e., HTML documents) from the Web sites of information providers and stores the retrieved page in a local repository. As is described below, basic pieces of information ("PSI" or

"attribute-value pairs") are derived from the stored Web pages. An attribute of a product refers to an essential or necessary property or characteristic of the product (e.g., size, color, memory, frequency, or weight). Each derived attribute may be characterized by an appropriately denominated value (e.g., a frequency value in MHz, a color value of "red").

Prior to extraction of attribute-value pairs from the Web pages retrieved by the crawler 250, the indexer 254 indexes the contents of the retrieved pages by operating upon various intermediate representations thereof. In particular, an HTML abstract syntax tree, full-text and structure index are developed based upon each retrieved HTML document. As is discussed below, the extractor 258 extracts PSI from an internal data representation 318 (FIG. 5) of the semi-structured HTML documents created by the indexer 254 in accordance with a predefined set of extraction rules. Each extraction rule defines a "target", a "method" and optionally a "projection", each of which are described below. In this regard the term "semi-structured" refers to an arrangement that as a whole does not have a precise structure, but within which certain elements have meanings based on their locations or surroundings within the arrangement.

The categorizer 262 is designed to identify a category within which a given PSI will be placed within a hierarchical framework, or "classification tree", maintained within the database 160. The classification effected by the categorizer 262 can be either completely automatic, or semi-automatic, with human validation of the possible categories being suggested by the classification algorithm. The database server 150 physically stores each categorized PSI within the disk systems or the like comprising the platform for the database 130.

Once a classification tree relevant to a particular category has been developed, potential buyers may invoke a product search engine to identify the most suitable vendor from which to purchase a particular item. The product search engine runs on the controller, and acts as a Web server by operating upon the central database 160 (via the database server 150) in response to search queries from potential buyers. Once a particular vendor has been selected, a shopping agent assists buyers by "pre-filling" forms to expedite any ensuing commercial transaction.

STRUCTURE OF CENTRAL DATABASE

As mentioned above, the database server 150 is operative to extract and categorize information from vendor web sites in particular product areas. One distinguishing feature of this operation is the introduction of structure into the retrieved information, which is accumulated from generally unstructured HTML documents. The introduction of such structure is intended to facilitate comparison shopping by standardizing price and product information. In addition, the structured information within the database 10 can be used to simplify online purchases by avoiding the need for repetitive data entry into online order forms.

In a preferred embodiment, the PSI that are manipulated by the database server 150 comprise a list of attribute-value pairs in which each value (e.g., text, numbers, URL, or other kind of information) provides a relevant characterization of a given attribute. For example, in the context of a product description the value corresponding to the attribute of "price" could be expressed in terms of an amount (e.g., a floating-point number) of a particular currency. Similarly, a value for attributes relating to product appearance could comprise a URL providing a link to a picture or other visual representation.

FIG. 4 is a block diagram depicting the logical relationship among the software objects comprising a preferred implementation of the object-oriented market model 210. In this preferred implementation, the market model 210 is structured so as to enable representation of various parameters associated with the market for a category of product or service. Referring to FIG. 4, a category object 270 includes a list of the various product categories (e.g., "laptop computer") for which market representations have been incorporated within the market model 210. A provider object 271 contains a set of known providers (e.g., "IBM") for each product category enumerated within object 270. The various product lines (e.g., "Thinkpad") associated with each provider are contained within a product line object 272. In addition, a model object 273 lists the various models (e.g., "600") for each product line registered in object 272.

In many instances a given model of product may be sold in a number of configurations. For example, an "IBM Thinkpad 600" laptop computer may be sold in configurations characterized by different quantities of RAM, different processor speeds, and perhaps with different packages of software applications. This aspect of the retail market for various product models is reflected by a configurations object 274. Each model configuration represented within the configurations object 274 may

be characterized by one or more attributes (e.g., "CPU", "RAM") retained within an attributes object 275. A value (e.g., "300 MHz", "128K") associated with each such attribute is preferably included within a "values" object 276.

In a typical retail transaction a consumer may be said to purchase a particular product "offering" when purchasing a specified good or service for a certain vendor. In a preferred implementation each potential product offering is defined by a product configuration, vendor, and price, and is retained within an offerings object 277. As is indicated by FIG. 4, potential vendors are listed within a vendors object 278. The vendors object 278 maintains data on vendors with fields such as name, contact information, payment preferences, type of business, goods sold, and order form formats.

A list of system users is maintained within a users object 282. Each user is characterized in terms of attributes contained within a user attributes object 283, with corresponding attribute values being maintained within a values object 284. In a preferred embodiment the users object 282 maintains data on users (i.e., potential buyers) relating to name, address, credit card number, phone number, ID number, social security number, electronic mail address, credit history, past system usage, and other biographical and usage information. This information is obtained when the user first registers with the system, or immediately prior to posting an initial bid for a selected product.

Current prices for product configurations are contained within a prices object 279. Prices denominated in various different currencies may be represented in terms of a common currency within object 279 upon being normalized in accordance with conversion information stored within currencies object 280. Upon purchase by a user of a given product offering, an identifier (e.g., an assigned user number) corresponding to the user is recorded within a transactions object 281 and a baskets object 287 is updated to reflect purchase of the offering. The baskets object 287 contains a list of the offerings, from potentially different vendors, which have been selected for purchase by the user.

In accordance with one aspect of the invention, the states of bids and watches created in the manner described below with reference to FIGS. 12 and 13, respectively, are maintained within a bids object 285 and within a watches object 286, respectively. In particular, the bids object 285 contains a list of any bids posted by a user on various models or offerings. The watches object 286 is structured to

contain a list of any watches posted by a user on various categories, models, or offerings.

FIG. 5 is a logical diagram representative of a preferred application architecture of database 160. In accordance with this preferred architecture, the database 160 is realized as an object-oriented database comprised of multiple, distributed services. Each service performs a specific function, and preferably communicates using TCP/IP-based protocols to enable execution on separate processors. This is intended to facilitate concurrent access to the database, to enhance scalability, and to facilitate the execution of sub-second searches on very extensive collections of categories, products, and offerings.

In a preferred implementation the logical relationships among the objects within the database 160 are computed by the database server 150 during loading of the database 160, and are then stored within the database 160 together with the data for the constituent objects. These stored logical relationships include "containment" relationships (i.e., "A has a B"), "type" relationships (i.e., "A is of type B", or "A is a subtype of B"), as well as other relationships commonly employed in object-oriented systems. This arrangement contrasts with conventional relational database systems, which typically do not provide for such pre-computation and storage of the relationship between objects. That is, in such conventional systems these relationships are generally computed "on the fly" in connection with the processing of a database query. The arrangement of the present invention is believed to be particularly advantageous in the case of very large databases, since such "on the fly" computations within large relational databases generally requires significant computational resources and may degrade performance.

The database 160 is also preferably structured to permit it to be updated with newly acquired market information while simultaneously, and without diminishing performance, allowing for the processing of user queries submitted via the Web server 100. Attainment of this objective is effected through the provision of separate input and output channels to and from the database 160, respectively. In the preferred implementation, these channels comprise independent processes executing on separate processors. Each process will preferably be designed to be cloned to any arbitrarily large extent, and will be disposed to independently communicate with the database server 150. In the representation of FIG. 5, the primary logical input channel

is identified as the load server 291 and the primary output channel is designated as the read service 292.

The load server 291 is responsible for providing the typically constant stream of data acquired from vendor web sites to a server control module 293. As was described with reference to FIG. 5, a configurations object 274 maintains a record of the various configurations in which a given model of product may be sold. In response to each categorized PSI received from the categorizer 262, the load server 291 determines whether each such PSI should give rise to a new product configuration or should simply update obsolete information contained within an existing configuration. The load server 291 makes this determination by testing for existence of identical PSI for the same model within the same category in the central database 160. If the new PSI is determined to correspond to a new product configuration, then the load server 291 informs the server control module 293 that a new entry needs to be created within the configurations object 274. In response, the server control module 293 creates the appropriate objects and stores them into memory and on disk. If the load server 291 determines the new PSI should be used to update obsolete information within an existing configuration, the server control module 293 finds an in-memory and on-disk references to the appropriate objects and updates these accordingly.

The read service 292 is responsible for processing user queries received by the Web server 100 and forwarded to the server control module 293. The read service 292 is a read-only application, and as mentioned above is disposed to answer user queries received from the server control module 293. In a preferred implementation such queries may contain model, vendor, offering or category related information. Search results provided by the read service 292 are ranked by relevance in the absence of other user-specified ranking criteria. In a preferred implementation the read service 292 incorporates a generalized, standardized dynamic parametric search engine of the type described in the following section.

As was discussed above, the categorizer 262 is designed to identify a category within which a given PSI will be placed within a hierarchical framework, or "classification tree", maintained within the central database. In order to enhance efficiency, prior to attempting to categorize any new PSI the categorizer 262 invokes a search server 290 (FIG. 5) to determine whether the new PSI has been previously categorized. Specifically, the search server 290 is a read-only device used by the

categorizer 262 to identify products already represented within the database 160. Queries concerning, for example, model names submitted to the server 290 return matching model names already existing within the database 160. In this way the categorizer 262 is able to become aware of which products are profiled within the database 160 based upon the results of queries made to the search server 290. Category information from the categorizer 262 is provided to the load server 291, which is a write-only application disposed to furnish this information to the database 160. The load server 291 is also responsible for computing the logical relationships between objects within the database 160 in the manner described above.

The update service 294 is a write-only application operative to write various elements of user information received through the inbound Web page service 130a (e.g., user profiles, preferences or product lists) to the database 160. The information stored in the database by virtue of operation of the load server 291 or update service 294 is maintained by an administrative service 295. In particular, the administrative service 295 furnishes reports summarizing the status of database 160, and performs administrative functions such has deletion, moving, or merging of categories.

The notification server 296 is a read-only application which receives information from the server control module 293 pertaining to the notification of users in connection with the occurrence of market events (e.g., a change in market price which results in a watch or bid priced being matched). This notification server 296 utilizes an e-mail processor 298 to send notification messages to users in the event of such a match.

PROCESSING OF USER QUERIES FOR PRODUCT AND SERVICE INFORMATION

As is discussed below, the information collected through the extraction and categorization processes described above enables the execution of dynamic parametric searches of the database 160 in response to user queries for product information. In this sense the term "parametric search" refers to a search based upon attribute values of objects of the type included within the database 160. For example, an attribute of an *IBM ThinkPad 600* laptop computer is "memory", and this attribute may assume values of 32MB, 64MB and the like. The parametric search engine identifies "memory" as an attribute of laptop computers, and allows users to search for models matching specific values of the "memory" attribute. A parametric search for laptop computers in which a value of 32MB for the attribute "memory" is

specified within the search query will return, inter alia, the result of *IBM Thinkpad 600*.

In the context of the present invention, the term "generalized parametric search" refers to a parametric search capable of being conducted across all object types and categories included within the database 160. Such a search may be effected using the preferred implementation of the database 160 described herein; namely, an implementation in which object attributes are abstracted in the same way across different types of objects, and therefore made available to a search process generally directed to such abstracted attributes (i.e., a search process not tailored for specific objects).

As used herein, the term "standardized parametric search" refers to a parametric search in which reference values may be defined for object attributes. Such definition of reference values is complicated by the fact that objects and products offered and advertised on the Web are characterized by non-standard attribute name and values. For example, in the case discussed above the "memory" attribute could be described as "RAM" by certain vendors. Similarly, the attribute value of "32MB" could be expressed as "32 megabytes" by certain vendors and as "32" by others. By providing reference values for any attribute names and values, matches to search queries may be identified independently of the Web-based originating source of the information matching the query.

In order to allow for the provision of reference values for attribute names, a set of aliases are developed for the sting chosen for each attribute name. In a preferred embodiment this aliasing process is effected on the fly during the categorization process. Specifically, an association is created between the string to be aliased and a reference string. For example, the attribute name "RAM" is associated with the reference string "Memory". In the same manner, the attribute value "extra large" is associated with the reference string "XL". Reference strings will preferably generally be presented to the user in lieu of vendor-specific information in order to maintain a coherent and simplified user interface. In the above example, "Memory" and "XL" would preferably appear in elements of the user interface (e.g., in headers, and pop-up menus), unless a need exists to display vendor-specific information.

During the search process query indexes are preferably developed based upon both reference values and vendor-specific information. This enables the same results to be returned irrespective of whether keywords in the query match the vendor-

specific terms or the reference strings. In other words, the vendor's lexicography is abstracted from the search queries, thereby allowing standardized parametric searches to be performed.

The aliasing mechanism of the present invention also facilitates inclusion of goods and services from international vendors within the database 160. Such inclusion is facilitated in one respect through the translation of reference values into a single common language. This translation enables parametric searches to be conducted without recourse to the language of original product descriptions provided by vendors. In addition, information may be extracted from HTML pages containing language different from the chosen common language by use of the "foreign" reference values. For example, if the chosen common language is English, then the reference value "*red*" could be employed to search Spanish-based HTML pages containing the attribute value "*rojo*".

The parametric search engine described herein is "dynamic" in the sense that new attributes and values of products are nearly automatically included within the relevant product categories by virtue of operation of the crawler 250, indexer 254, extractor 258 and categorizer 262. In a preferred implementation this update process proceeds without human intervention once an operator has validated a proposed category for a newly discovered product, and entered the appropriate reference strings for a newly discovered attribute or value. This aspect of the present invention facilitates maintenance of very large databases designed to support the generalized exchange of goods and services described herein.

It will typically be desirable that the search results returned in response to user queries are ranked in accordance with the relevance of each result to the subject query. For purposes of explanation, such search results are referred to below as "product models" or simply "models", since in the preferred embodiment the database 160 contains a categorized repository of information relating to the product models of various vendors. In a preferred implementation the most highly ranked models are those drawn from categories deemed to be most relevant to the search query. Within each category, the ranking of search results is predicated largely on the extent to which keywords within the search query appear in the names and descriptions of each product model. Both the determination of category relevance, and the relative ranking of models within and between categories, are described in further detail below.

Set forth below under the heading "Category Relevance" are preferred criteria utilized in determining the relevance of various categories to a given search query; that is, in determining those categories most likely to include a product or service sought by the user submitting the search query. In what follows distinctions are drawn between the various "levels" of a given categorical representation of a particular model. For example, a first-level category could be named "printer", and a second-level category (i.e., a subcategory of the first-level category) could be named "laser printer". The expression "cat1:cat2:laser:printer" is intended to reflect the relationship between the first-level category "laser" and the second-level category "printer". Similarly, a second-level category itself named "laser printer" would be represented by "cat1:cat2:laser printer", while a first-level category of the same name would be represented as "cat1:laser printer". The criteria below also refer to "stemmed keywords", which correspond those keywords which have been have been truncated or otherwise modified to eliminate plurals, "ing" postfixes and verb infinitives prior to being processed as search terms.

Category Relevance

(1) The existence of all stemmed keywords from a search query within a category name at a single category level. For example, after being "stemmed" all of the terms within the search query "laser printers" are present in the second level of the category representation "cat1:cat2:Laser Printer:xyz:cat4".

(2) The existence of all the actual keywords from a search query within a category name at a single category level. For example, the exact terms within the search query "laser printers" are present in the second level of the category representation "cat1:cat2:Laser Printers".

(3) The existence of a word in a category name of similar meaning to a keyword within a search query. For example, the similarity of the keyword "pen" to the category name "pencil".

(4) The existence of a single keyword within a category name. For example, the presence of the word "pen" in a search query and in a category name.

(5) The existence of a single stemmed keyword within a category name. For example, the presence of the stemmed keyword "pen" (e.g., derived from the search query term "pens") within the category name "pen".

(6) The word position and/or character position of a stemmed keyword within a category name at a given category level. For example, the stemmed keyword "pen" appears in the second word position, and the fourth character position, at the third level of the category representation "cat1:cat2:xyz pens:cat4".

(7) The relative depth at which a stemmed keyword appears within a category name of a multi-level category representation. For example, in the four-level category representation "cat1:cat2:pens:cat4", the "depth" of the stemmed keyword "pen" therein may be characterized as "2" (i.e., the penultimate category level).

(8) The ratio of the number of product model names within the category containing the stemmed keyword to the total number of product models within the category.

Once category relevance has been established in accordance with the above criteria, relative weights are assigned to the models within each category in accordance with the criteria set forth below under the heading "Model Weighting Criteria". Prior to this assignment of relative weights, those categories having model names containing any stemmed keywords from the search query are identified. Within each identified category, the various models are then weighted based upon the Model Weighting Criteria (presented in decreasing order of significance).

Model Ranking Criteria

(1) The number of stemmed keywords present in the category name and relevant product fields of the subject model. The relevant product fields are typically those maintained by the provider object 271 and product line object 272 (e.g., product name, brand, and product descriptions). In a preferred implementation it is initially determined whether all stemmed keywords are included in the applicable category name. If so, then all models included within such category are assigned the maximum relative weight. For example, processing of the query "laser printer" would result in all models in a category named "cat1:cat2:laser:printers" being assigned the maximum relative weight.

(i) If not all stemmed keywords from the search query are contained in the applicable category name, then the only models selected from the category are those for which the remaining keywords are contained in an associated descriptive field. As an example, consider the case in which the user query is "hp laser printer". In this instance the only models selected (i.e., assigned a nonzero relative weight) within the category named "cat1:cat2:laser:printers" are those containing "hp" in an associated field. The relative weight assigned to each selected model is computed based upon the number of such remaining keywords contained in associated fields; that is, models associated with fields containing larger numbers of stemmed keywords are assigned higher weights.

(ii) If none of the stemmed keywords are contained in the applicable category name, then the only models selected from the category are those for which the remaining models keywords are contained in an associated descriptive field. Again, the relative weight assigned to each selected model is computed based upon the number of such remaining keywords contained in fields associated with a given selected model.

(2) The relative order and position of the stemmed keywords within a category name. This criterion is used to distinguish between models having names which appear equally relevant to the name of a general category, but which have different relevance to the names of more specific categories. In a preferred implementation the following three criteria are used in combination to determine the relative weights among models within a given category:

(i) The correspondence between the order of the stemmed keywords in the query and in the category name. For example, in the case of a query containing the string of stemmed keywords "laser printer", models associated with a category name containing "laser" prior to "printer" would be weighted more highly than models within a category having a name containing "printer" before "laser".

(ii) The extent of correspondence between all words in a category name and the stemmed query keywords. For example, in the case of a query containing the string of stemmed keywords "laser printer", models associated with a category representation having the exact string "laser printer" at one category level would be weighted more highly than models within a category level named "laser printer supplies".

(iii) The depth of the category level within a category representation at which the correspondence described in (i) and (ii) above occurs between stemmed keywords and category names. Specifically, the "deeper" within a category representation which such correspondence occurs, the higher the weights assigned to models associated with the particular category level. For example, in the case of a query containing the string of stemmed keywords "laser printer", models associated with a category "cat1:cat2:laser printer" may be accorded higher weights than those included within a category named "cat1:laser printer:cat2".

Overall Ranking

As discussed above, the relevance of applicable categories to a given search query is initially determined in accordance with the Category Relevance Criteria. In a preferred implementation the models within each such category are preliminarily weighted based upon the relevance of their respective categories. Each model within a pertinent category is then assigned an additional weight based upon application of the Model Ranking Criteria. The aggregate weight assigned to each model provides one basis for determining the ranking of models within the search results presented to a user in response to submission of a search query. This ranking may be modified in accordance with other user-defined criteria (e.g., price, brand). As an example, in connection with query "laser printer" considered above, a user could specify that all "Canon" printer models be more highly ranked than "HP" models and that all printer models priced at less than $500 be more highly ranked than higher-priced models.

UPDATING OF THE CENTRAL DATABASE

Crawling

FIG. 6 provides an illustrative representation of the distributed architecture of the crawler 250. The crawler 250 includes one or more crawl engines, each of which is usually comprised of a plurality of crawl agents 302 designed to retrieve Web pages from the web sites of vendors and other information providers. The crawl agents 302 are observed and controlled by log clients 304 and by a control panel 305, respectively. Typically, one instance of a log client 304 and one instance of a control panel 305 is assigned to any human operator designated to monitor operation of the crawler 250. Each control panel 305 is generally configured to control (e.g., start, stop, pause) operation of some or all of the crawl agents 302 associated with the crawler 250.

In a preferred embodiment the crawl agents 302 are designed to be capable of traversing "dynamic" Web sites (e.g. sites relying on CGIs, ASP, or any other Web server technology functioning as a gateway facility to content publishing software). The crawl agents 302 should also be capable of updating the retrieved Web pages stored within a crawler page repository 306, and be capable of support of large-scale crawling activity. A set of rules is preferably associated with each site in order to identify those URLs that will be scanned by the applicable crawl agent 302. In FIG. 6, this set of rules is represented by a customization module 307. That is, each such set of rules specifies the HTML documents that are to be downloaded from each site by the applicable crawl agent 302. Such rules will generally be in the form of subtree definitions (i.e., URL prefixes), or pattern definitions.

Each crawl agent 302 is preferably capable of "rewriting" the URL for each retrieved page and storing the rewritten URLs in a local URL repository 310. Specifically, each crawl agent 302 uses conventional regular-expression rewriting rules to rewrite the URLs actually associated with the retrieved pages. This rewriting allows for efficient handling of "session-based" URLs, normalization of complex URLs, and the like. The URL repository 310 also reflects the current "status" of each URL stored therein. For example, the status of a given URL could indicate that its associated Web page is to be "crawled" at a specified future date, or could identify the last date the page was crawled.

As an example of URL normalization, consider that certain Web servers do not recognize a distinction between uppercase and lowercase characters within URLs. In this case the URLs for the Web pages from such servers are normalized to include either all uppercase or all lowercase characters. This reduces the bandwidth required by the crawl agents 302, since it is unnecessary to download capitalization information for each individual URL character.

In the case of session-based URLs, crawling efficiency is improved through the encoding of session IDs within the applicable URLs. In particular, URLs may be of the form *http://www.site.com/some/path/to/an.asp?sessionID=12345ABC* where *12345ABC* is the session ID representative of a particular "session". The actual numeric value of the session ID is generated each time a user enters a session-based Web site, and may be valid only for a predetermined session period. However, a crawl agent302 may be incapable of scanning an information intensive Web page prior to expiration of the applicable session period. In order to avoid being forced to download the page being scanned due to such session expiration, the crawl agent 302 formulates a canonical version of each session-based URL encountered during the scanning process. That is, each session-based URL is rewritten using a fixed ID. In an exemplary implementation this formulation is effected as follows:

rewrite "sessionID=[0-9A-Za-z]+\(&?\)" "sessionID=XXXXXXXX\1"

rewrite "&sessionID=[0-9A-Za-z]+" "&sessionID=XXXXXXXX"

where the first argument of each *rewrite* expression corresponds to a pattern to be matched within the original URL, and the second argument corresponds to a string within the rewritten URL replacing this pattern.

In order to access all pages of interest at various web sites it may be necessary to perform certain actions upon form-based interfaces and the like. Each crawl agent 302 will thus preferably be designed to ascertain when a form does not actually require textual input from a user, and will then compute all potential ways of activating the form (e.g., through various menu options). In general, forms containing only combinations of the following elements do not require textual input from a user:

<INPUT TYPE=hidden>
<INPUT TYPE=checkbox>
<INPUT TYPE=radio>
<INPUT TYPE=submit>
<SELECT> </SELECT>

Forms containing only combinations of the above elements can only generate a finite number of input requests in response to user selections, since only a finite number of user selections are possible. On the contrary, a form containing a <TEXTAREA> or an <INPUT TYPE=text> may generate an infinite number of requests, since the requests are based upon arbitrary textual input from a user or crawl agent 302. Since many forms containing only combinations of the above elements also invoke a POST method, it is desirable that each crawl agent 302 also support this method.

The crawler 250 is preferably designed to efficiently extract product and service information from vendors' Web sites by scanning only selected pages. For example, the crawl agents 302 will preferably be disposed to scan only those pages on the Web site of a product vendor which pertain to relevant product information. This differs from conventional crawling approaches in that all pages lacking product information are bypassed, while dynamically generated and other subsidiary pages (ignored by conventional crawlers) are downloaded to the extent product information is present. An exemplary set of rules for focusing the crawling operation only upon pages likely to contain informative product or service information is set forth below:

```
explore subtree "/catalog
        download all URLs beginning with /catalog
explore pattern "product"
        download all URLs containing the "product" string
```

Extraction

FIG. 7 illustratively represents the process by which the indexer 254 and extractor 258 operate upon HTML documents (i.e., Web pages) stored within the crawler page repository 306. The indexer 254 and extractor 258 are designed to collectively extract structured information from a Web page composed of arbitrary HTML content. This is achieved by applying expressions of an *extraction language*, site-specific extraction rules 310, and built-in strategies in order to filter unnecessary content from the retrieved HTML documents 312.

In a preferred embodiment the extractor 258 applies "regular expressions" on limited, kinded character strings to extract elementary information (i.e., attribute values). In this sense "limited" strings are those containing only data relevant to a given attribute, and "kinded" strings are configured such it is possible to discern to which attributes the string relates. For example, the string "*32MB EDO SDRAM*" is limited in scope to the attribute "memory". In contrast, the string "*166MHz Pentium*

MMX processor with 2GB hard drive and 20XCD-ROM drive" is neither limited nor kinded. The term "regular expression" is intended to have the meaning ascribed to it in Mastering Regular Expressions.; Jeffrey E. F. Friedl; O'Reilly (1997).

The extraction process contemplated by the invention also takes advantage of the characteristic of the HTML structure to inherently provide "limited strings". In particular, many attributes included within an HTML document are delimited by HTML elements such as tables, lists, descriptions, paragraphs, or <img> separated strings. Accordingly, in order to extract an entire set of attributes for a given PSI the HTML structure is traversed and such limited strings are identified. Although the HTML structure (or available markup information such as XML) is relied upon to a certain extent to facilitate the extraction process, the structure of Web pages can be expected to change somewhat over time. Nonetheless, the efficiency of the extraction process is enhanced by addressing portions of a Web page in accordance with its expected structure and contents.

As was mentioned above, several intermediate representations of the retrieved HTML document are manipulated during the process of extracting relevant attribute-value pairs. Each of these intermediate representation may be expressed in the form of a data structure. One such data structure, created by the indexer 254, consists of a full-text and structure index. This index is preferably largely of the form described in "An algebra for structured text search and a framework for its implementation"; Charles L. A. Clarke, G. V. Cormack, and F. J. Burkowski; *The Computer Journal*, 38(1):43–56, 1995. This reference includes a description of an algebra for computing intervals in semi-structured documents. The relevant interval operators are:

- _ *is in* ([), *is not in* (![), _
- *contains* (]), *does not contain* (!]) _
- *conjunction* (&), *disjunction* (|) _
- *followed by* (...)

The "base intervals" within a given HTML document are specific to an associated application. In a preferred embodiment a set of base intervals having the following query syntax are utilized:

word: a word denotes the location intervals of all its occurrences in the document

%num: denotes the location intervals of all numbers in the document

<e>: denotes the location intervals of all HTML tags of element <e> (e.g., <a> denotes the intervals of all anchors). In addition, <comment> denotes all the comment intervals.

#n: denotes all intervals of size *n*

%line: denotes all line intervals

Another intermediate representation of a retrieved HTML document, termed an HTML abstract syntax tree ("AST"), is generated by the parser 314 within the extractor 258 (FIG. 7). In a preferred embodiment the AST for each retried Web page is patterned after ASTs utilized in the field of compilers (see, e.g., Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: principles, techniques, and tools.* Addison-Wesley, 1986.). The relevant specific features of these trees are:_ weak typing. Although HTML is normally defined as an SGML DTD, a practical implementation of the parser 314 must be robust enough to handle syntactically incorrect documents, and still produce an AST. Thus, the typing constraints given by the DTD are not always respected in the AST. Comments may be an important information for extraction purposes, and must thus be included in the tree locations. Each node in the AST is annotated with the locations it spans in the original document (e.g. starting and ending locations, in character offsets from the beginning of the document). As is described below, a virtual tree is constructed from selected portions of the original AST. During this process the indexes are instructed to return results relevant to a given virtual tree.

In a preferred embodiment each retrieved HTML document undergoes the following extraction process:

1. The HTML document is parsed by the parser 314, which also constructs an AST based upon the parsed HTML document. An internal data representation 318 comprised of the AST and the index created by the indexer 254 are stored within the extractor 258 (FIG. 7).

2. The applicable extraction script is parsed.

3. Virtual pages are constructed by a rule engine 320 based in part upon the AST within the internal data representation 318.

4. For each virtual page, the pertinent extraction rules are sequentially applied by the rule engine 320 and the extracted information is collected.

5. In connection with the application of each rule by the rule engine 320,

a) a target (defined below) is identified,

b) all relevant rewriting rules are applied to the target, and

c) the extracted attribute-value pairs are collected and provided to the categorizer 262 as pieces of structured information.

The extraction rules referenced in the above process description are each composed of a target, a method, and a projection. The target may be viewed as an anchored path, which corresponds to an anchor and an associated navigation path within the AST. The navigation path is composed of directives to move laterally and longitudinally within the AST, with possible additional constraints. An anchored path denotes zero, one or several nodes in the AST. Each anchor is defined by an element type and a query for the index, and corresponds to a node in the AST which: (i) is of the given element type, and (ii) spans the applicable location interval. An initial result of the query for the index is a location interval in the original document. The content of each node within the AST consists of all the text (i.e., HTML PCDATA and CDATA) associated with the node.

The method for each extraction rule defines precisely how attribute-value pairs are constructed using each target. A projection is an optional constraint within an extraction rule applicable to the attribute that is desired to be extracted using the rule.

The present invention contemplates several methods for extracting attribute-value pairs from HTML documents using the intermediate data structures described above. A first extraction method is employed when the identified subpart of the HTML structure contains the value of an attribute. This method is frequently used for the model and price attributes of a product. In this method, the textual contents of the target are passed through the various rewriting rules to extract attribute values. In a second method only the URL of the selected node (assumed to be either <IMG>, <A> or < FORM>) is returned as the field value. A third extraction method contemplates use of a "built-in strategy". The principal object of each such built-in strategy is to attempt to map any inherent structure within original HTML file into a more descriptive structure. For example, it may be attempted to create a descriptive structure of the form "<DL><DT><DD>...</DL>.". In this example each "<DT>"

segment contains text recognizable as an attribute name, and each "<DD>" segments contains text providing an attribute value.

Another such built-in strategy involves manipulation of those "<SELECT>" elements present in an HTML page. The various "<OPTION>" entries in a <SELECT> element define a potential set of values of the <SELECT> element. Accordingly, the <OPTION> entries can be extracted as potential values of an attribute (e.g., color, size) defined by a given <SELECT> element.

Yet another built-in strategy of the present invention is applicable to "vertical tables" defined by "<TABLE>" elements. Any exemplary vertical table is set forth below:

Specifications	
Resolution	1280 x 1024
LCD Screen	1.8 "
Memory	8 MB

With the exception of the first row, each row of the exemplary table may be said to define an attribute-value pair. For example, the value "1280 x 1024" is seen to be associated with the attribute "Resolution". An extraction procedure modeled upon this strategy would operate upon each row of the table so as to return all defined attribute-value pairs.

An additional built-in strategy contemplated by the present invention may be termed a "description" strategy. This strategy is applied to the "<DL>" elements within a "<DL>" description in the same way as the "vertical table" strategy is applied to "<TABLE>" elements. That is, an extraction procedure modeled upon the strategy would extract as many attribute-value pairs as there are "<DT>"-"<DD>" pairs within the applicable <DL> description.

The extraction of relevant attribute-value pairs from an exemplary HTML document in accordance with the present invention will now be described with reference to FIGS. 8 and 9. The exemplary HTML document is depicted in FIG. 8, which is seen to include table descriptions of two different models of Olympus digital cameras; namely, the D-620L and the D-400Z models. FIG. 9A illustratively represents the HTML content corresponding to the table description for the D-620L

model included within the HTML document of FIG. 8. As was discussed above, the indexer 284 is operative to create a full-text and structure index of a retrieved HTML document prior to initiating extraction of attribute-value pairs. In this regard FIG. 9B provides a representation of a portion of such an index for the HTML content of FIG. 9A. As was also discussed above, in the preferred embodiment an AST and virtual trees are created prior to proceeding with extraction of attribute-value pairs. In this regard FIG. 9C illustratively represents a portion of the AST for the HTML content of FIG. 9A

Once the data structures of FIGS. 9B and 9C have been generated, an extraction script is executed in the manner described below. The extraction script includes a specification of all virtual trees to be used during the process of extracting relevant attribute-value pairs from the subject HTML document. In a preferred implementation each virtual tree is derived from the AST, and includes a row <tr> from the AST referencing a <form> construct. FIG. 9D shows an exemplary of one of the virtual trees capable of being derived from the AST of FIG. 9C. In this case the applicable extraction script would include the following portion directed to derivation of the virtual tree of FIG. 9D:

```
product =
   shared matching * . <title> + shared matching * . <h1>
   + matching <tr>("<tr> ] <form>")
```

Once the indexes, AST and virtual trees for a given HTML document have been generated as explained above, the extraction script proceeds to apply a set of extraction rules to the virtual trees defined by the script. In a preferred implementation each extraction rule specifies the information to be extracted from a given field of a particular virtual tree node. For example, the following extraction script excerpt mandates that the link (i.e., URL) of the <img> node in the first cell (cells are numbered from zero) of the specified row be extracted. The script returns a "(photo, link)" attribute-value pair, where "photo" corresponds to the name of the field containing the extracted URL.

```
fields *
    from link photo
    on node * . <tr> . <td>(0) . <img>
```

The following excerpt of extraction script defines a rule pursuant to which the textual contents from the second cell of the specified row are extracted. A "(model, text)"

attribute-value pair is returned, where "text" refers to the textual content from the field "model".

```
fields *
 from content model
 on node * . <tr> . <td>(1)
```

The rule defined by the following extraction script applies the "vertical table" strategy discussed above to the virtual tree of FIG. 9D. The script returns the product specification information contained within the "table" construct of this virtual tree.

```
fields *
 from strategy vertical_table
 on node * . <tr> . <table>
```

Execution of the following script excerpt results in extraction of two attribute-value pairs through application of regular expressions to the textual content of the <h1> node. The regular expression applicable to the "provider" field extracts the first word of content. The remaining content is extracted through application of another regular expression to the category field.

```
fields *
 from direct
   provider = "\([^ ]\)" -> 1;
   category = "[^ ][ ]+\(.*\)" -> 1;
 on node * . <h1>
```

Tables I and II below depict the results of applying the extraction script predicated on the rules discussed above to the "D-620L" and "D400Z" models of Fig. 8, respectively.

TABLE I

Attribute	Value
link	http://www.vendor.com/images/products/d620l.jpg
model	D-620 L
resolution	1280x1024
LCD screen	1.8"
memory	8MB
provider	Olympus
category	Digital Cameras

TABLE II

Attribute	Value
link	http://www.vendor.com/images/products/d400z.jpg
model	D-400Z
resolution	1280x960
LCD screen	1.8"
memory	8MB
provider	Olympus
category	Digital Cameras

The extraction process of the present invention may also be applied recursively across multiple linked HTML pages. Such a recursive process is employed due to the fact that all the information required to fully describe a given object may not be available from a single HTML page of a Web site. For example, category information may be provided on a different HTML page than other product information. In a preferred implementation the procedures inherent within the extraction scripts described above are applied to a subset of the HTML pages of a given Web site. This subset is aggregated by recursively traversing through the HTML of the Web site via the hypertext links embedded in each page. The following clauses are representative of an exemplary set of actions performed during the process of aggregating such a subset of HTML pages:

```
when url "list.cfm" {
product = matching <a>("link:'detail'");

fields *
... from content provider
   on node * . <a>;
include * . <a>;}
```

In the preceding example, relevant content from an HTML page having a URL matching the regular expression "list.cfm" is initially extracted in the manner described above. When processing a page having a URL matching the regular expression "list.cfm", the clause

```
product = matching <a>("link:'detail'"),
```

defines the virtual trees associated with the HTML page. In this example, each virtual tree is composed of a single "<a>" node, and the number of virtual trees is equivalent to the number of anchors having URLs containing the "detail" word. In connection with each virtual trees, information for fields in the provider object 271 is obtained from the "content provider" field at node * . <a> through the extraction of textual content from the <a> node. As a consequence of the rule "include * . <a>" recursive extraction then proceeds through pages linked by this anchor. Extraction of additional product information from the remainder of the HTML page of this example would then proceed in accordance with other extraction rules present in the applicable script. This additional information, together with the information obtained for the provider object 271, would collectively comprise the PSI extracted from the HTML page.

The preceding example illustrates the manner in which a subset of HTML pages relevant to the extraction process may be traversed via established hyperlinks through application of predefined expressions. As was discussed above, the present invention contemplates the use of built-in strategies during the extraction process as a means of mapping any inherent structure within an original HTML file into a more descriptive structure. In this regard certain built-in strategies may call for the application of "domain extraction rules" on the contents of a node of a virtual tree. In a preferred implementation such domain extraction rules comprise certain of the extraction rules described above; that is, extraction rules based upon the application of regular expressions. As is described further below, the set of such regular-expression based extraction rules pertinent to a given "domain" is specified by a domain definition. The collection of the domain definitions for each domain of interest are represented as the domain specifications 324 (FIG. 7).

Each domain definition preferably consists of a set of extraction rules, each of which is preferably of the following form.

```
rule name =
        trigger = extended_regexp
```

```
[unless = extended_regexp]
extracts fieldname [rewrite_rules]+
[extracts fieldname [rewrite_rules]+]
;
```

where descriptions are provided below for the terms "trigger", "extended_regexp" and "rewrite_rules". Each extraction rule of a domain definition is applied in accordance with the extraction script for the applicable built-in strategy . The application of an extraction rule involves application of its constituent rewrite rules to the textual contents of a virtual tree node targeted by the extraction script. For a given targeted node, all rewrite rules of the extraction rule are applied in order to yield a list of attribute value pairs.

The term "extended_regexps" refers to a syntax for facilitating combination of standard regular expressions with logical connectives AND and OR, and are of the following form:

```
extended_regexp ::=
      R "regexp"
   | RL "regexp"
   | OR ( extended_regexp ; ... ; extended_regexp )
   | AND ( extended_regexp ; ... ; extended_regexp )
```

The following are examples of the "rewrite_rules" referred to above:

```
[=]"regexp" -> n
[=]"regexp" -> constant_string
[=]"regexp" -> R "rewrite_string"
```

In a preferred implementation a rewrite rule is always applied to the text content of a node. As is indicated above, the leftmost portion of each rewrite rule is a regular expression which is invoked to search the text data a single time. Unless the symbol "=" precedes the "regexp" in the rewrite rule, the subsequent search of the text data is case insensitive. The rightmost portion of each rewrite rule gives the result of the rule.

In the first exemplary rule set forth above, the result of "n" corresponds to the matching group for the regular expression (i.e., a zero value for "n" means that the entire string defined by the regular expression should be matched, a unity value for "n" means that the first group defined by the regular expression should be matched, and so on). Within the second exemplary rule, the term "constant string" is the identifier for the character string returned by the rule (e.g., the string "Formatted" is

returned by this type of rule in the practical example below). In the case of the third exemplary rule, the result is obtained by performing a rewriting operation using the "rewrite_string"; that is, by conventionally substituting "\1", "\2", and so on with their corresponding matching groups from the original string.

A "trigger" expression is a regular expression serving to trigger application of an extraction rule from a domain definition. Each trigger expression will typically contain one or more keywords from the domain of information associated with a given product attribute. An "unless" expression is a regular containing exceptions to the trigger expression. The purpose of an unless expression is to prevent ambiguity within the keywords of a trigger expression from leading to an incorrect application of the associated extraction rule. For example, consider the case in which "Intel" is a keyword for the attribute "processor" in an extraction rule directed to the domain of personal computer hardware, and in which "EtherPro" is a keyword in this rule's unless expression. Since "EtherPro" is the name of a commercial network card, this unless expression will prevent the extraction rule from being incorrectly applied to descriptions of computer networking equipment such as network cards.

As a practical example of the foregoing, set forth below are domain specifications pertinent to descriptions of diskette storage devices:

```
rule capacity =
  trigger = OR (RL "[0-9\.]+[ ]*mb"; RL "[0-9\.]+[ ]*gb")
  extracts capacity
    "\([0-9\.]+\)[ ]*mb" -> R "\1 MB"
;
rule size =
  trigger = OR (RL "\"")
  extracts size
    "3 1/2\"" -> "3 1/2\""
    "3.5\"" -> "3 1/2\""
    "5 1/4\"" -> "5 1/4\""
    "5.25\"" -> "5 1/4\""
;
rule format =
  trigger = OR (R "DS"; R "SS"; R "DD"; R "HD"; RL "format")
  extracts format
    "\bSS\b" -> "SS"
    "\bDS\b" -> "DS"
    "\bDD\b" -> "DD"
    "\bHD\b" -> "HD"
    "\bformatted\n" -> "Formatted"
    "\bunformatted\n" -> "Unformatted"
;
```

Categorization

FIG. 10a illustratively represents an exemplary sequence of processes performed by the categorizer 262. A primary objective of the categorizer 262 is to identify the category in which a given PSI will appear within the database 10. As is indicated by FIG. 10a, in a step 320 the extraction process described in the previous section produces a PSI needing to be categorized. In a preferred embodiment the categorization process is initially attempted to be effected using a process hereinafter referred to as "reverse vendor category mapping" (step 322). If this "reverse mapping" process is successful, the PSI is stored within the market model 210 of database 160 in accordance with its assigned category (step 324). If the process is unsuccessful, categorization is attempted to be performed using a "clustering" process (step 326), as is generally described in Automatic text processing: the transformation, analysis, and retrieval of information by computer; Gerard Salton; Addison-Wesley (1989).

In a preferred implementation, the reverse vendor category mapping process (step 322) contemplates extraction of certain structural information pertaining to categories from the vendor's Web site. This extracted structural information is then mapped to the category object 270 of the market model 210 within the database 160. The extracted structural information is generally in the form of strings uniquely identifying vendor-defined categories for the vendor's products. This structural information may frequently be extracted from the names of vendor-defined categories, and is often found on the product pages of the vendor's Web site (i.e., the HTML pages containing the most significant product information). In a preferred implementation, the extraction process is commenced from an HTML page at the vendor's site which contains links to all other pages containing category information (including the relevant "product pages). Using the extraction process described above, category information from the linked pages is then gathered and temporarily stored until those pages containing the most detailed product information have been scrutinized. Using this temporarily stored information, the categorizer 262 associates each vendor-defined category with a category defined by the category object 270. These associations are then stored within the category object 270 for subsequent re-use when obtaining updated or new product information from the vendor's site for inclusion in the market model 210 or other portions of the database 160.

In certain cases, vendor-defined category information may be unavailable or of a sufficiently regular format to be extracted and associated with the categories of the category object 270. In other instances the granularity of the extracted category information is not appropriate for such association with the category object 270. For example, a vendor may carry both desktop and laptop computers and include them both within a category entitled "Computers". If the most general categories relating to computers within the category object 270 are "Laptops" and "Desktops", then the vendor's category "Computers" is equally applicable to both and no association is created. A vendor's site may also contain apparently mistaken product categorizations (i.e., the classification of products unrelated to computers within the vendor-defined category "Computers"), in which case a mapping between the vendor-defined category and the category object 270 is not effected. In such cases where category mapping is inappropriate, categorization is preferably achieved through the use of clustering.

In the clustering approach (step 326), each category is represented by a vector (step 328) obtained as a linear combination of the words in the category name and the words in all PSI currently known to belong to this category (that is, all words in the attributes of the PSI that are considered relevant). Each PSI is also represented by a vector obtained as a linear combination of the various relevant text attributes (such as model name, short and long description). In all the linear combinations, the respective weights of the different components are configurable; in particular, the respective weights of the vectors forming a PSI representation are vendor-specific, since they depend on the quality and length of the vendor provided information. Due to the inheritance structure of the category tree, each category vector is obtained by summing the parent category vector and the vector representing the category specific information (step 330).

For each PSI, the cosine distance of its vector representation from the vector representations (e.g., C1, C2 in FIG. 10b) of all known categories is computed. Classification can then proceed automatically by assigning the PSI to the category separated by the smallest cosine distance from its vector representation (step 332). Alternately, the classification process may involve human validation of categories suggested as being close to the PSI of interest based upon such computation of cosine distance (step 334). Upon such validation, the PSI is stored within the market model 210 of database 160 in accordance with its assigned category (step 336). During the

classification process categorizer 262 learns the words included within a newly classified PSI, thus refining the vector representations for each category. Again, the exact words selected to be "learned" will vary among vendors due to the varying quality of information provided by different vendors.

The extractor 262 will also preferably be capable of recognizing the brand of a product even in cases where this parameter cannot be directly extracted as an attribute. In a preferred implementation the extractor 262 will employ an algorithm to extract brand information from an input string of provider data derived from a given HTML document. Specifically, the input string is searched to determine whether it includes any known brand. If this search is unsuccessful in identifying a product brand, a regular expression identified as "provider hints" is applied to the input string in an additional effort to ascertain whether brand information is included.

SHOPPING AGENT

In a preferred embodiment a shopping agent is made available to potential purchasers via the central server 100. The shopping agent is designed to simplify the purchase process by capturing user information and automatically completing forms required to process transactions. Upon identifying a product, a potential purchaser is presented with a list of known vendors and given the option of making a selection. If a selection is made, the user purchases the product directly from the vendor through a consistent interface provided by the shopping agent. The shopping agent is capable of discerning the requisite customer information required by vendors' purchase order, and enters any customer information saved from previous transactions into the standard interface presented to the purchaser. The purchaser then enters any other required information, and the information within the completed order form is automatically passed to the Web site of the vendor. All subsequent communication, such as order confirmation and shipment information, occurs whenever possible directly between the vendor and the buyer. However, in cases where the vendor-supplied information to be provided to the buyer is of a state which cannot be made compatible with the buyer's interface, the shopping agent will function to proxy the entire transaction. In the general case, the shopping agent relieves purchasers from having to complete order forms from scratch in connection with each purchase from a Web-based vendor. The shopping agent also advantageously streamlines the management of user accounts and passwords across vendor sites.

The shopping agent preferably acts as a proxy in the transaction, rewriting the HTML pages on the fly (pre-setting form values to known user-specific information). The value to users of the transaction proxy feature is twofold: it provides users with a single electronic order form for all Web purchases, and it streamlines the completion of order forms for repeat users. The use of Web-based proxies is described generally in "World wide Web proxies", Ari Luotonen, *Proceedings of the First International Conference on the World-Wide Web*, 1994.

FIG. 11 is a flow chart representative of the processing performed by the shopping agent of the present invention during interaction with a potential purchaser. After selecting a product to be purchased from a particular vendor, a user clicks on a shopping agent icon or the like displayed upon a buyer interface (step 350). In response, a request is transmitted to the central server 100 (step 354) and forwarded to the central database 10 (step 358). The request is received at the central database 10 (step 362), and values are retrieved for all fields for which information has been previously stored (step 366). As discussed above, vendor information is stored within database by the database server 150 (step 370) as a consequence of the crawling, indexing and extraction operations described above (collectively represented by step 374). Accordingly, during step 366 any information pertinent to the vendor specified during step 350 is retrieved from the database 10 and provided to the central server 100. In like manner buyer information collected during previous transactions involving the shopping agent is also stored within the database 10 (step 378) and provided to the server 100 during step 366.

In step 382, the central server 100 receives the field values retrieved from the database 10 during step 366. The field values are placed within the applicable page defined by the shopping agent, and the modified shopping agent page is returned to the user (steps 386 and 390). In addition, the central server 100 uses the field values to at least partially complete any transaction form acquired from the vendor site (step 394). The transaction form is then returned to the user (step 398) and inspected (step 402). It is then determined whether any further user interaction with the transaction form is required to complete the contemplated transaction (step 406). If so, the user continues the transaction by entering any other required information, or by selecting among available preferences, options or the like (step 414). The server 100 receives this additional information and/or selections (collectively, "requests") (step 418) and

forwards the requests to the vendor site (step 422). Processing then proceeds in the manner described above with reference to step 394.

Upon completion of a given transaction (step 406), the shopping basket for the applicable vendor is updated (step 410) and the user is presented with the same shopping agent page returned in step 390. The user is then prompted to decide whether to purchase additional products from the vendor (step 430). If no additional products are to be purchased, the user indicates this intention by clicking an "exit" icon (step 434). If the user desires to purchase additional products, the user select s a "buy" icon (step 438) and the central server 100 receives a corresponding "buy" request (step 442). This request is transmitted by the central server 100 to the vendor site (step 446), at which point it is determined whether the vendor cart has been filled (step 450). If not, the step 446 is repeated for a predetermined number of trials. If the maximum number of trials is reached and the vendor cart has not been filled, the user is informed that the transaction cannot be recorded by the vendor. If it is determined that the vendor cart has been filled, the central server 100 retrieves the first page of the vendor's transaction form (step 454) and presents this page to the user (step 402).

Referring to FIG. 11, processing by the shopping agent commences at step 438 with respect to each vendor from which the user anticipates purchasing a product. In the typical case, the processing sequence associated with each vendor will begin at step 438 and terminate at step 410. Within this processing sequence, a loop defined by steps 442, 446, 450, and 454 serves to fill a particular vendor cart with items selected by the user. A "transaction" loop (i.e., the loop including the steps 402, 406, 414, ..., 402) facilitates consummation of a transaction with a particular vendor site, and also terminates at step 410.

BID, MARKET WATCH, AND MATCH PROCESSES

As was discussed above with reference to FIG. 11, the shopping agent of the present invention provides a list of vendors offering a product identified by a user. If the user desires to obtain a lower price or otherwise purchase the product on terms not offered, the present invention provides a facility for the user to submit a bid for the desired product. By allowing users to submit bids for desired products, purchases can occur at prices below those published by vendors. Lower-priced purchases may also be effected by aggregating bids submitted by users and negotiating with potential vendors.

FIG. 12 is a flow chart depicting an exemplary manner in which bids are placed by users, and evaluated by vendors, in accordance with the present invention. In a step 480, the user receives a list of vendors offering a desired product at the conclusion of a product search performed by the shopping agent (FIG. 10). If, for example, the lowest market price ("LMP") offered by any vendor listed by the shopping agent exceeds the price the user is willing to pay, the user may elect to submit a bid for the desired product (step 484). If the user has previously registered (step 488), additional information may be requested (step 492) and the corresponding user profile updated (step 496). Otherwise, a registration process is initiated during which pertinent demographic and other information is obtained (step 500).

Following completion of steps 492 and 500, the user may be asked to enter into a "clickwrap" agreement or the like providing that the user will become obligated to pay a brokerage fee should any bid submitted by the user be accepted by a participating vendor (step 502). After assenting to this agreement, the user enters credit card information (step 504) and a bid for a desired product (step 506). It is then determined whether the bid is less than the LMP (step 508). If not, the user is informed that at least one vendor is currently offering the desired product at a price less than the amount of the user's bid (step 510). If the bid exceeds the LMP, a credit card authorization process is performed (step 512). If authorization is obtained, the bid is accepted by the system in the sense of being qualified to present to potential vendors (step 516). The user is informed if the credit card authorization process is unsuccessful (step 510).

Bids accepted by the system operator are assigned a code in order to ensure anonymity of the associated user (step 518), and are posted in database 10 with an associated date/time stamp (step 520). In step 496, the user profile is updated to

record the posting of the newly accepted bid. When it is determined that a bid has been satisfied or has expired (step 526), the bid is cancelled (step 528) and the corresponding user profile is appropriately updated (step 496).

Upon posting of an outstanding bid within database 10, the LMP for the product identified by the posted bid is refreshed (step 530). The outstanding bid is compared to the refreshed LMP (step 532), and remains posted within database 10 if it is determined to be less (step 534). Bids exceeding the corresponding refreshed LMP are deemed to be satisfied, but brokerage fees are preferably not imposed in this situation (step 536). In this case the user is simply notified of the existence of a LMP below the amount of the previously outstanding bid (step 538).

When a vendor decides to accept an outstanding bid (step 540), the vendor becomes bound to deliver the identified product at the bid price by manifesting assent through electronic or other means (e.g., via facsimile) (step 542). Upon receiving the vendor's assent, the outstanding bid is deemed satisfied (step 544) and the corresponding vendor and user profiles are updated (steps 542 and 496, respectively). The user is then charged the applicable brokerage fee, which in a preferred implementation comprises a percentage of the difference between the LMP and the amount of the accepted bid (step 546). Notification, and instructions relating to completion of the transaction with the accepting vendor, are then provided to the user (step 548).

FIG. 13 is a flow chart depicting an exemplary sequence of steps involved in implementing a market watch utility of the present invention. In a step 560, the user receives a list of vendors offering a desired product at the conclusion of a product search performed by the shopping agent (FIG. 10) and ascertains the LMP. The user may then elect to initiate a market watch process (step 562), at which point it is determined whether the user has previously registered (step 564). If not, the user completes a registration process (step 566); if so, the user may be prompted to enter additional information (568). In both cases the corresponding user profile is updated (step 569). The user then identifies a product and an associated market price ("watch price") of interest to the user (step 570). It is then determined whether the watch price is less than the LMP (step 572), and if not the user is prompted to revise the watch price (step 574). Once a watch price below the LMP has been entered, the user is assigned a code to ensure anonymity (step 576).

Outstanding watches are posted in database 10 with an associated date/time stamp (step 578). In step 569, the user profile is updated to record the posting of the newly accepted bid. When it is determined that a watch has been satisfied or has expired (step 582), the watch is cancelled (step 584) and the corresponding user profile is appropriately updated (step 569). Upon posting of an outstanding watch within database 10, the LMP for the product identified by the posted watch is refreshed (step 584). The relevant watch price is compared to the refreshed LMP (step 586), and remains posted within database 10 if it is determined to be less (step 588). Watch prices exceeding the corresponding refreshed LMP are deemed to be satisfied (step 590), and the user is notified (step 592).

FIG. 14 provides a flow chart representation of the processes occurring during a bid matching process of the present invention. In a step 600, the user receives a list of vendors offering a desired product at the conclusion of a product search performed by the shopping agent (FIG. 7) and ascertains the LMP. If the user wishes to purchase the desired product from a vendor other than the vendor currently offering the LMP, the user may request that the preferred vendor agree to sell the desired product at the LMP (step 604). That is, the user may submit a request that the desired vendor match the LMP currently being offered by a different vendor. It is then determined whether the user has previously registered (step 606). If not, the user completes a registration process (step 608); if so, the user may be prompted to enter additional information (step 610). In both cases the corresponding user profile is updated (step 612). The user then submits the match request by selecting the preferred vendor from a list of participating vendors (step 614). A code is assigned to the match ensure anonymity (step 616), and the match is posted in the database 10 with its assigned code and date/time stamp (step 618). In step 612, the user profile is updated to record the posting of the newly accepted bid. When it is determined that a match has been satisfied or has expired (step 622), the match is cancelled (step 624) and the corresponding user profile is appropriately updated (step 612).

If the match has not been satisfied or expired, the market price from the desired vendor ("MPDV") for the product of interest is refreshed (step 624). The outstanding match is compared to the refreshed MPDV (step 626), and remains posted within database 10 if it is determined to be less than the refreshed MPDV (step 628). Matches exceeding the corresponding refreshed MPDV are deemed to be satisfied, but brokerage fees are preferably not imposed in this context (step 630). In this case

the user is simply notified that the MPDV is less than the amount of the previously outstanding bid (step 632).

When a vendor agrees to sell the preferred product at the price specified in the match (i.e., at the LMP) (step 634), the vendor becomes legally bound to deliver the preferred product at this price by manifesting assent through electronic or other means (step 636). Upon receiving the vendor's assent, the outstanding match is deemed satisfied (step 638) and the corresponding vendor and user profiles are updated (steps 620 and 612, respectively). The user is then charged a commission fee (step 640). Notification, and instructions relating to completion of the transaction with the accepting vendor, are then provided to the user (step 642).

Although the above application has been described primarily in the context of a particular implementation, one skilled in the art can readily appreciate that the teachings of the present invention may be applied to other contexts and implementations. Thus the application is meant only to be limited by the scope of the appended claims.

What is claimed is:

1. A computer-based method for facilitating a transaction between a buyer and a vendor, said method comprising the steps of:

compiling a database containing information relating to a plurality of product or service offerings;

entering a bid into a computer, said bid identifying one of said offerings within said database and an associated offer price;

making the bid available to one or more vendors of said offering;

entering into the computer, in response to the bid, an acceptance from one of said vendors; and

notifying the buyer of said acceptance from said one of said vendor.

2. The method of claim 1 wherein said step of compiling includes the step of performing an internet-based search for said information.

3. The method of claim 2 wherein said step of performing an internet-based search includes the step of extracting vendor-supplied information from internet sites associated with said vendors.

4. The method of claim 3 wherein said step of compiling further includes the steps of assigning said vendor-supplied information to product categories, and of storing said vendor-supplied information within said database on the basis of said product categories.

5. The method of claim 1 further including the steps of performing a search of said database in response to a query supplied by said buyer, and of selecting said one of said offerings based upon results of said search.

6. A computer-based method for monitoring prices of product or service offerings, said method comprising the steps of:

compiling a database containing price information relating to said offerings;

entering a watch request into a computer, said watch request identifying a selected offering within said database and an associated watch price;

performing an internet-based search to determine vendor prices for at least said selected offering, said price information being updated in accordance with said vendor prices; and

notifying a user when said watch price is within a predefined range of at least one of said vendor prices for said selected offing.

7. The method of claim 6 wherein said step of compiling includes the step of performing an internet-based search for said information.

8. The method of claim 7 wherein said step of performing an internet-based search includes the step of extracting vendor-supplied information from internet sites associated with said vendors.

9. The method of claim 8 wherein said step of compiling further includes the steps of assigning said vendor-supplied information to product categories, and of storing said vendor-supplied information within said database on the basis of said product categories.

10. A system for facilitating a transaction between a buyer and a vendor, comprising:

a memory unit containing a database of information relating to a plurality of product or service offerings; and

a processor unit for

receiving a bid, said bid identifying one of said offerings within said database and an associated offer price;

making the bid available to one or more vendors of said offering;

receiving an acceptance from one of said vendors; and

notifying the buyer of said acceptance from said one of said vendor.

11. The system of claim 10 further including a searcher module for performing an internet-based search for said information.

12. The system of claim 11 wherein said searcher module includes an extractor for extracting vendor-supplied information from internet sites associated with said vendors.

13. The system of claim 12 further including means for assigning said vendor-supplied information to product categories, and for storing said vendor-supplied information within said database on the basis of said product categories.

14. In a system for facilitating transactions between buyers and vendors, a database of information relating to a plurality of product or service offerings, said database comprising:

an offerings object for providing a representation of said offerings;

a vendors object containing information relating to vendors associated with said offerings;

a prices object containing price information relating to said offerings; and

a configurations object including information relating to product or service configurations corresponding to said offerings.

15. The system of claim 14 further including a model object containing information relating to product models identified within said configurations object.

16. The system of claim 14 further including a product line object referenced by said model object, a provider object referenced by said product line object, and a category object reference by said provider object.

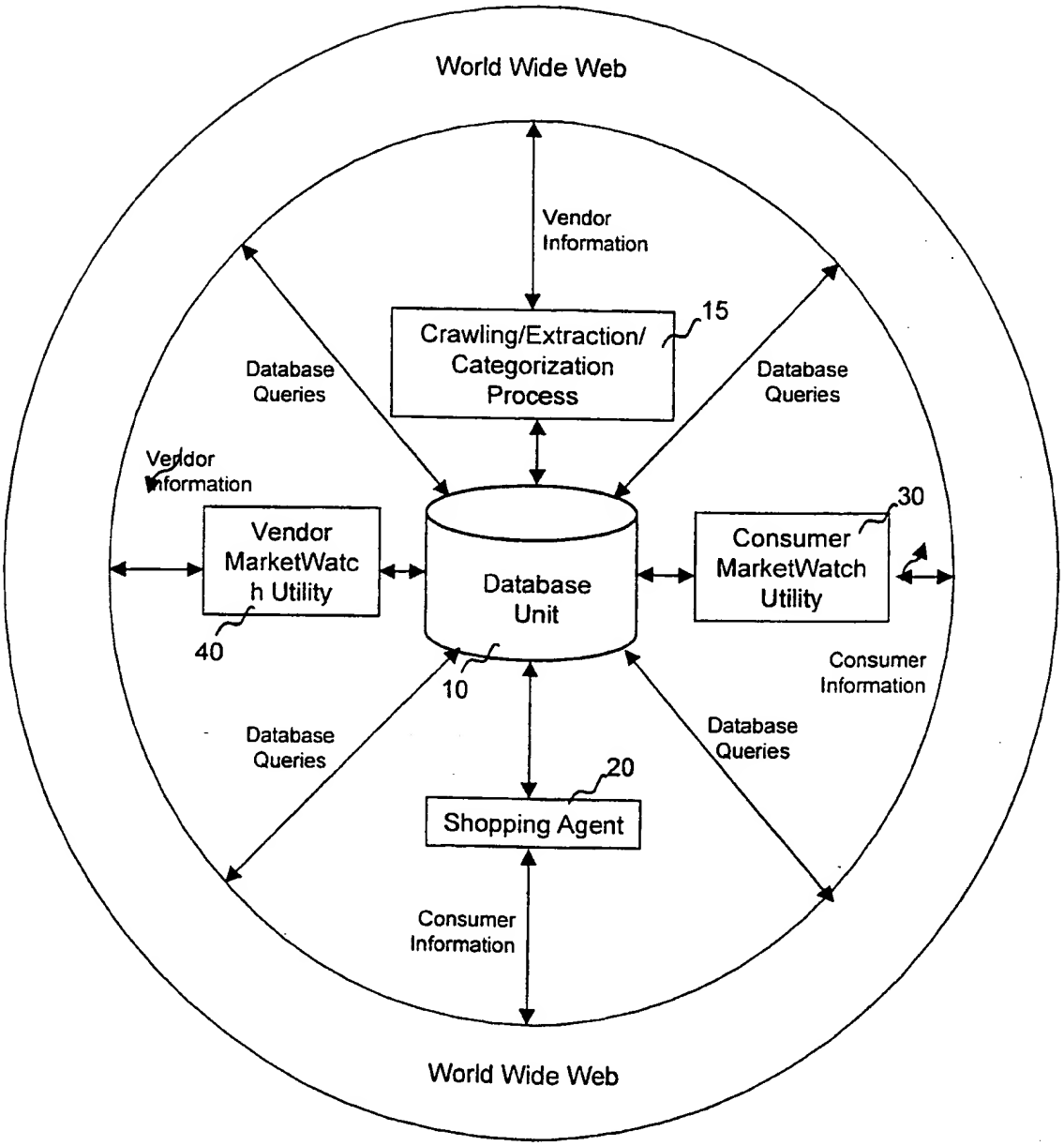

World Wide Web
Vendor Information
Crawling/Extraction/ Categorization Process
15
Database Queries
Database Queries
Vendor Information
Vendor MarketWatch Utility
40
Database Unit
10
Consumer MarketWatch Utility
30
Consumer Information
Database Queries
Database Queries
20
Shopping Agent
Consumer Information
World Wide Web

FIG. 1

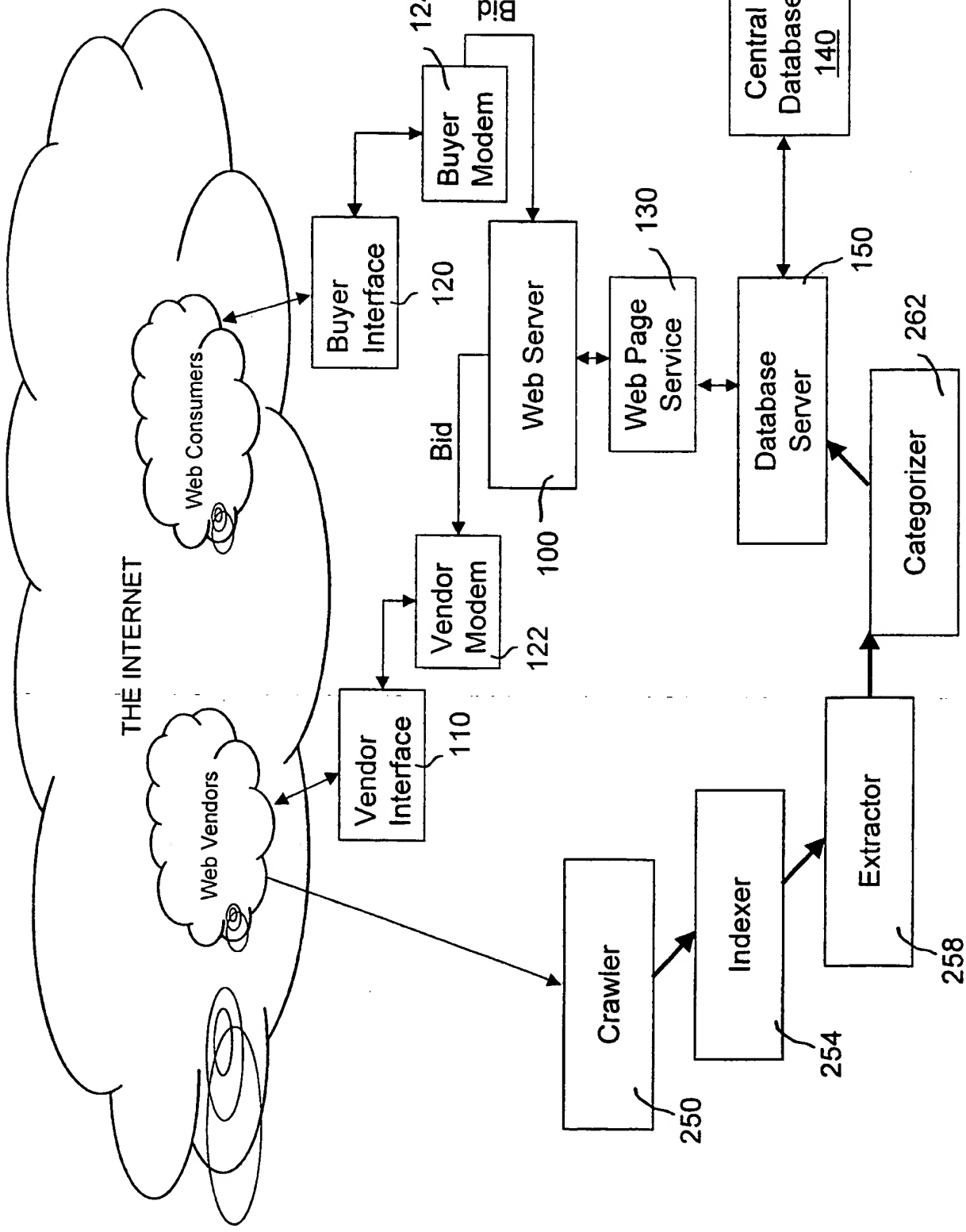

THE INTERNET
Web Consumers
Web Vendors
Buyer Interface
120
Buyer Modem
124
Bid
Vendor Interface
110
Vendor Modem
122
Bid
Web Server
100
Web Page Service
130
Central Database
140
Database Server
150
Categorizer
262
Extractor
258
Indexer
254
Crawler
250

FIG. 2

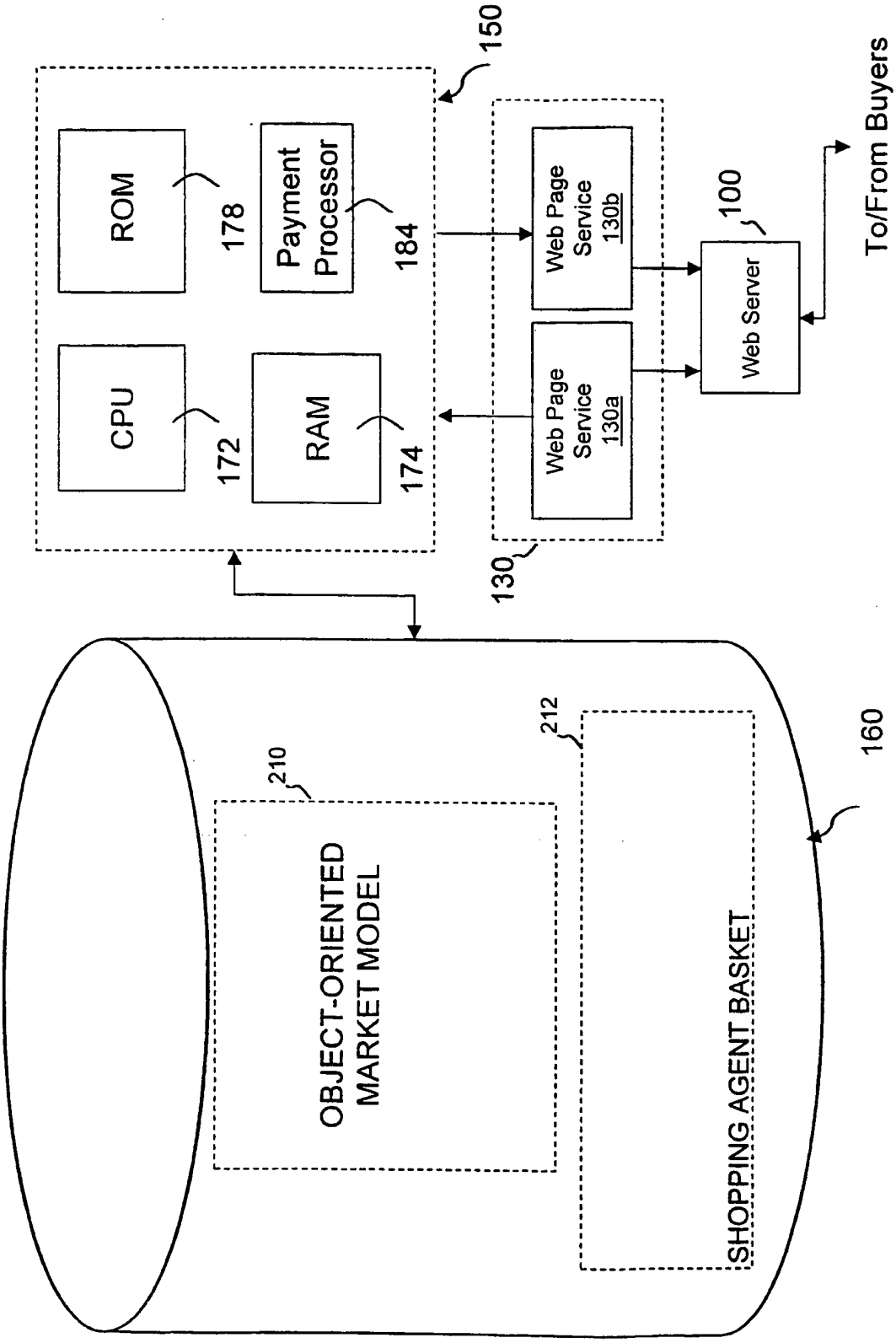

150
ROM
178
Payment Processor
184
CPU
172
RAM
174
Web Page Service 130b
Web Page Service 130a
130
100
Web Server
To/From Buyers and Vendors
160
210
OBJECT-ORIENTED MARKET MODEL
212
SHOPPING AGENT BASKET

FIG. 3

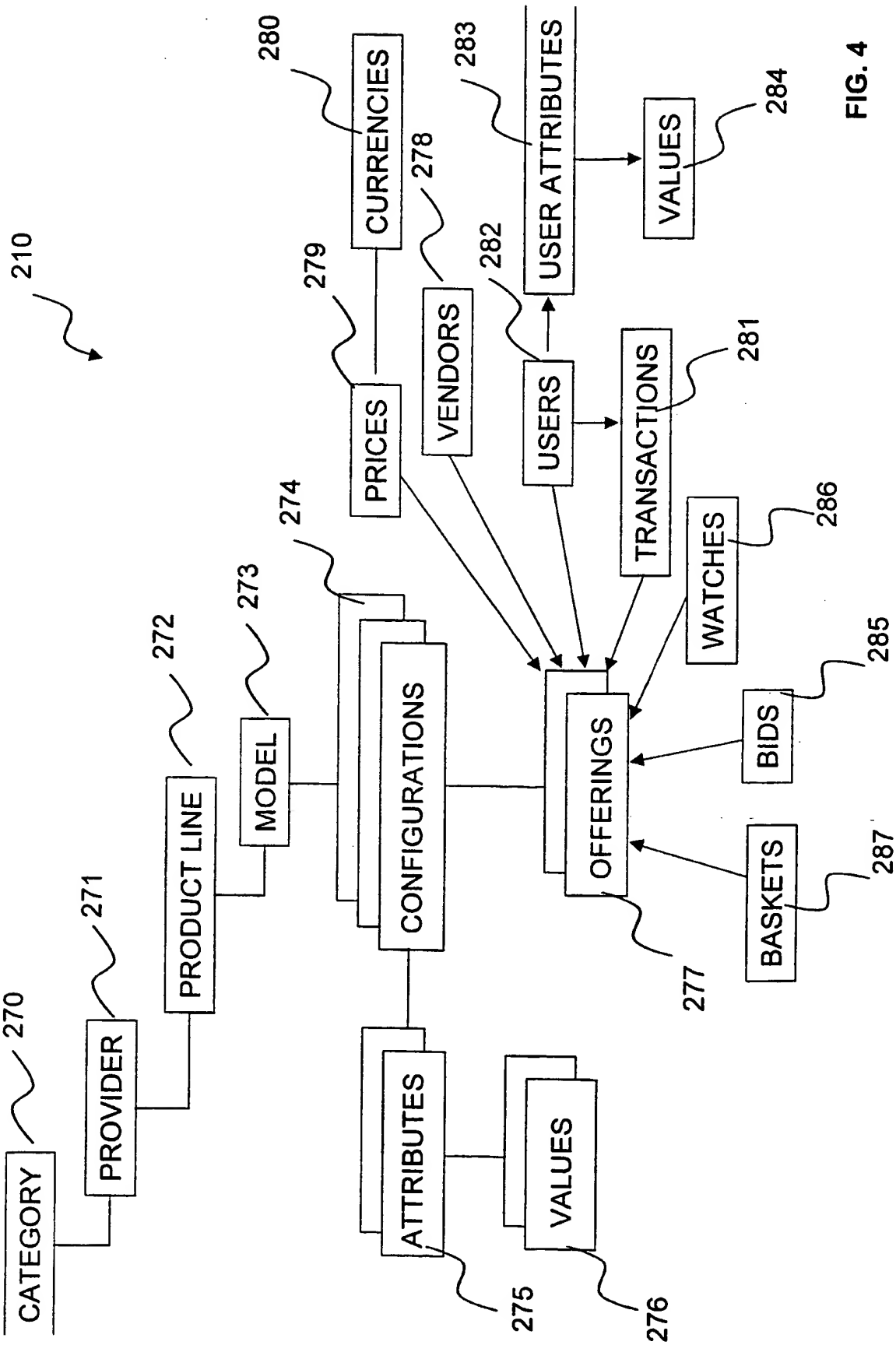

210
CATEGORY
270
PROVIDER
271
PRODUCT LINE
272
MODEL
273
CONFIGURATIONS
274
ATTRIBUTES
275
VALUES
276
OFFERINGS
277
PRICES
279
CURRENCIES
280
VENDORS
278
USERS
282
USER ATTRIBUTES
283
VALUES
284
TRANSACTIONS
281
WATCHES
286
BIDS
285
BASKETS
287

FIG. 4

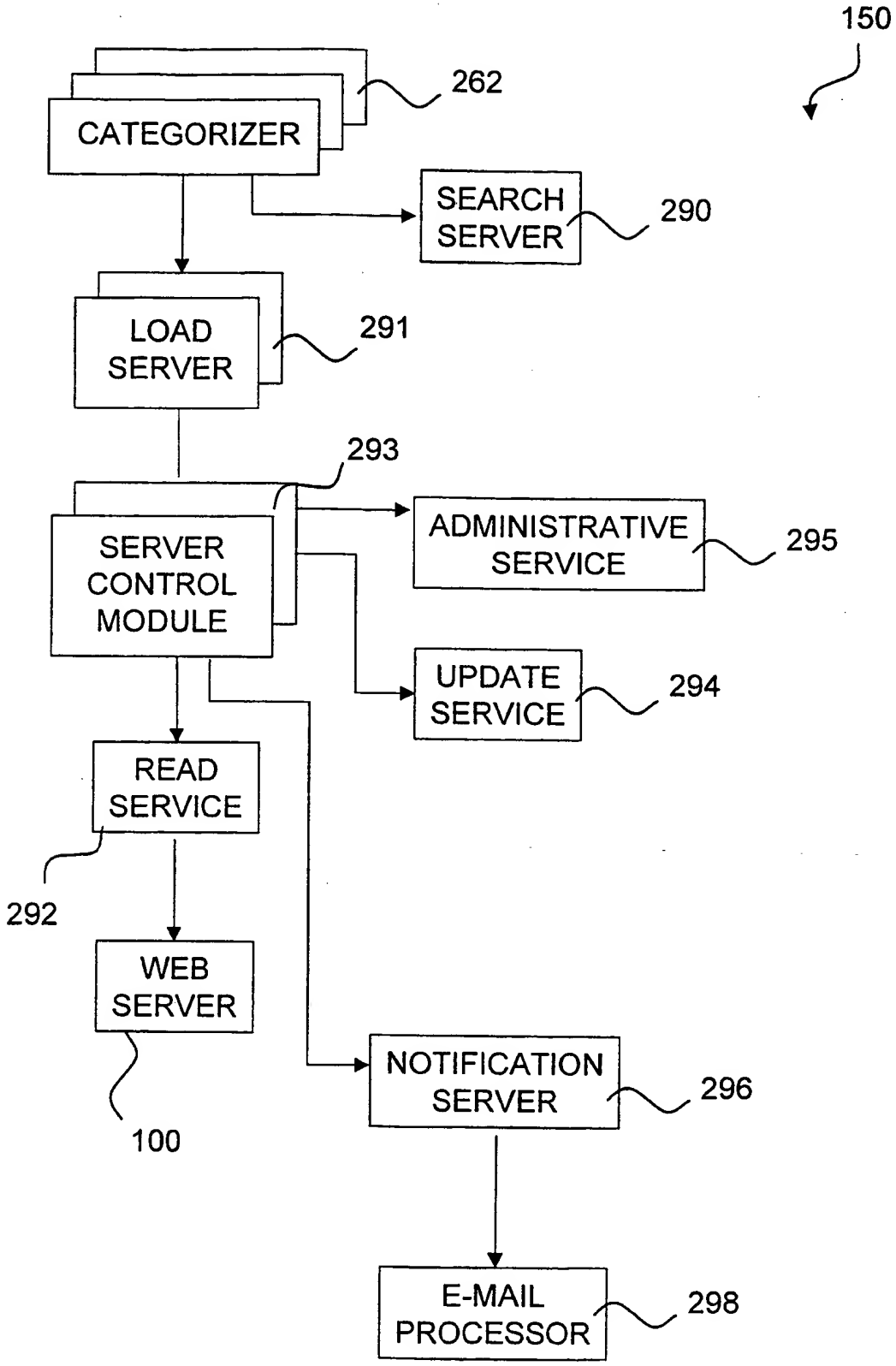

150
262
CATEGORIZER
SEARCH SERVER
290
LOAD SERVER
291
293
SERVER CONTROL MODULE
ADMINISTRATIVE SERVICE
295
UPDATE SERVICE
294
READ SERVICE
292
WEB SERVER
100
NOTIFICATION SERVER
296
E-MAIL PROCESSOR
298

FIG. 5

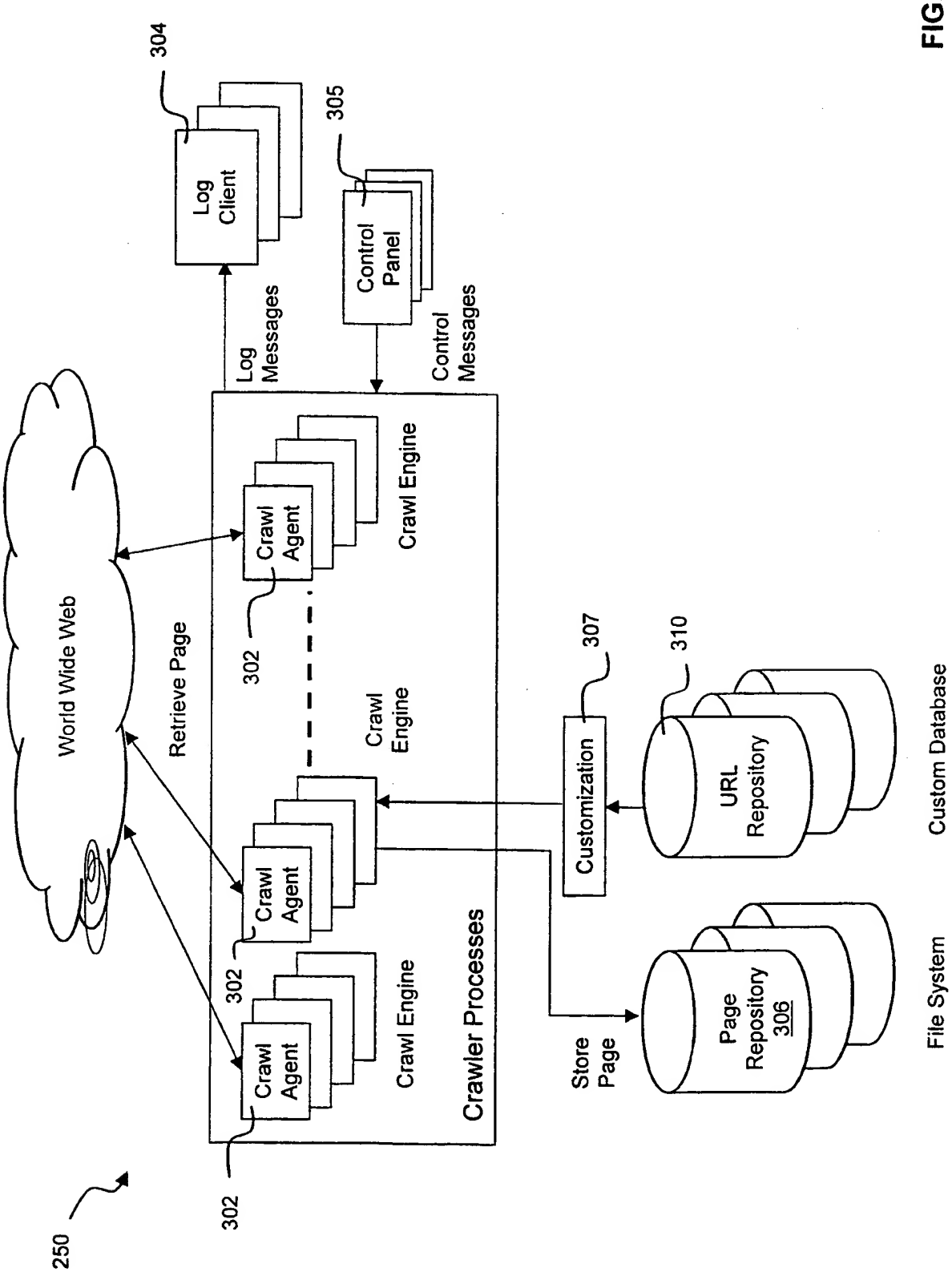

250
World Wide Web
Retrieve Page
Crawler Processes
Crawl Engine
Crawl Agent
302
Log Client
304
Log Messages
Control Panel
305
Control Messages
Customization
307
URL Repository
310
Custom Database
Page Repository 306
File System
Store Page

FIG. 6

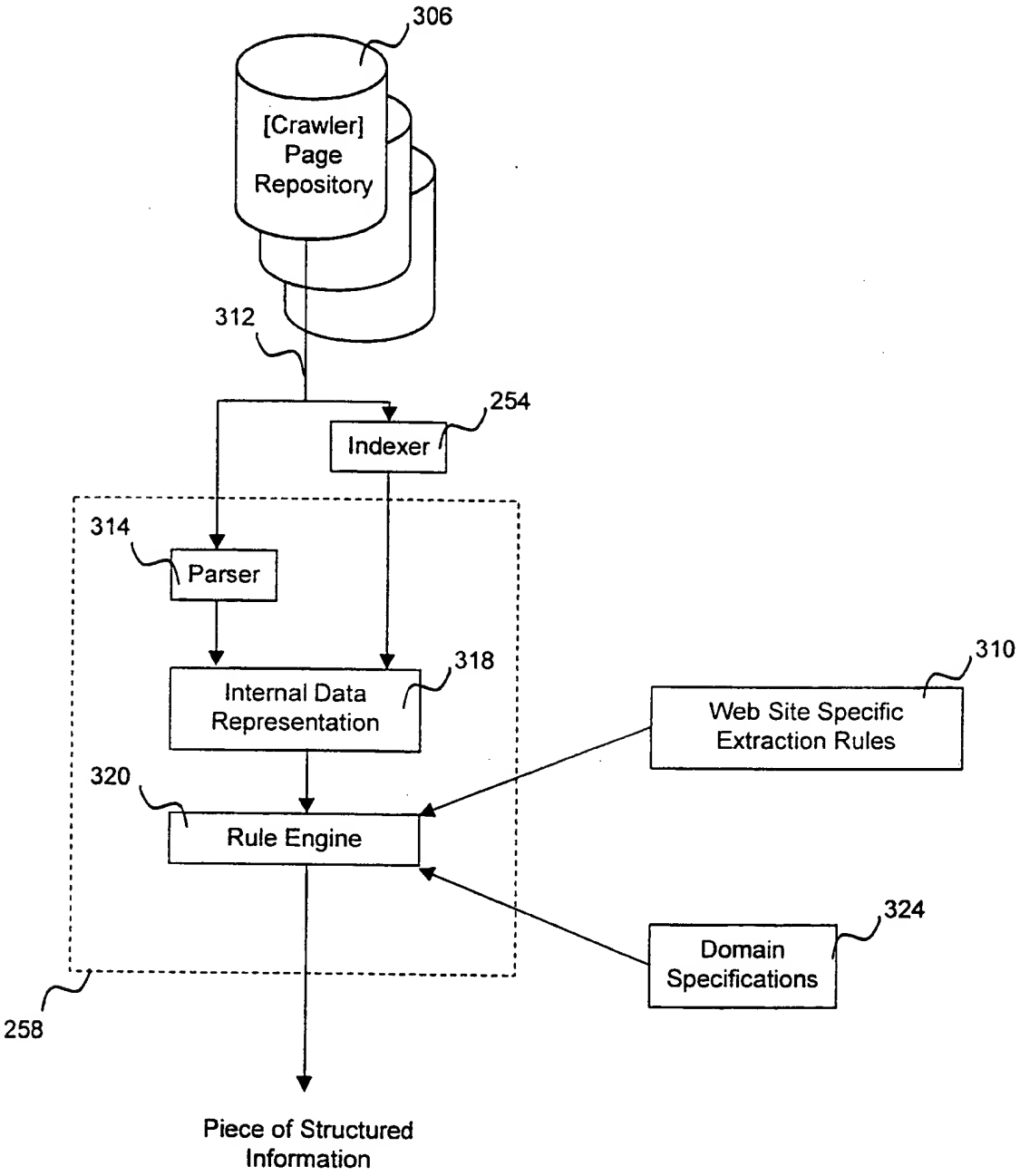

306
[Crawler] Page Repository
312
254
Indexer
314
Parser
318
Internal Data Representation
310
Web Site Specific Extraction Rules
320
Rule Engine
324
Domain Specifications
258
Piece of Structured Information

FIG. 7

Olympus Digital Camera

	Model	Description	Our price	
	D-620 L	True 1.4 million pixel resolution, a 3X zoom lens, and external flash sync support in a convenient ZLRTM (Zoom Lens Reflex) design. Specifications Resolution 1280x1024 LCD Screen 1.8" Memory 8MB	$ 1059.99	Order
	D-400Z	Features 1.3 Megapixel, 3X Optical Zoom, & Floppy Disk Compatibility 1.6" Specifications Resolution 1280x960 LCD Screen Memory 8MB	$ 799.99	Order

FIG. 8

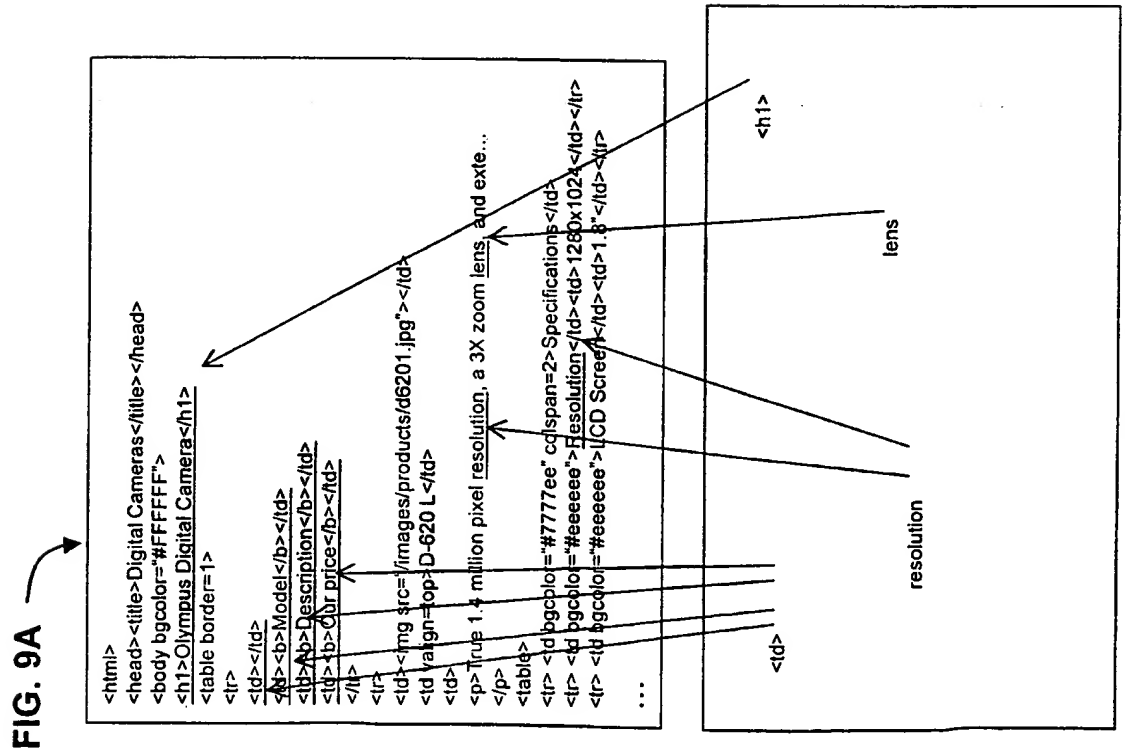


FIG. 9A

FIG. 9B

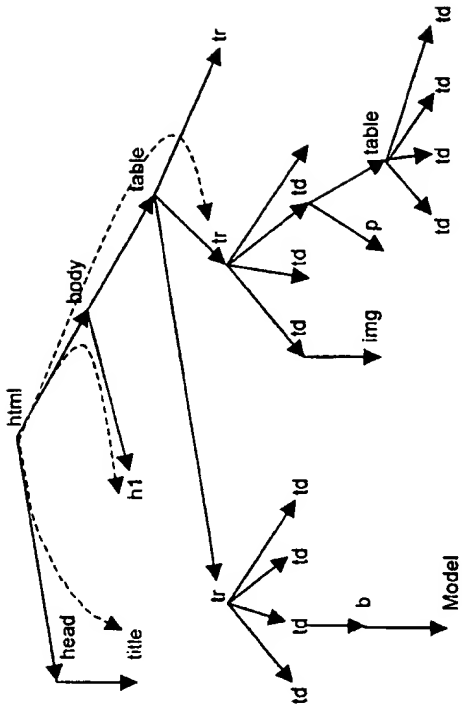


FIG. 9C

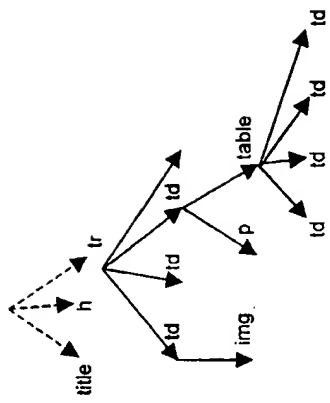


FIG. 9D

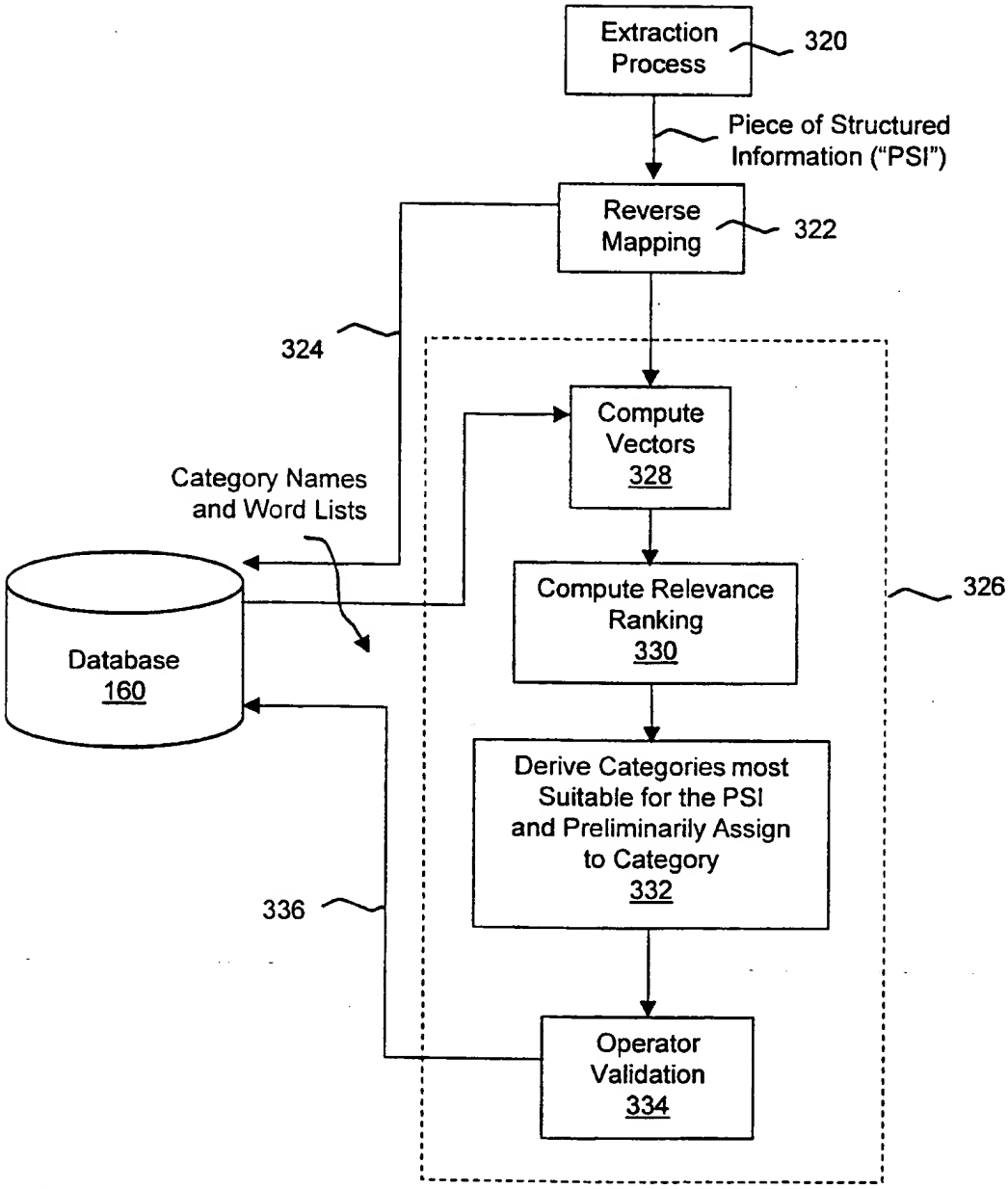


FIG. 10a

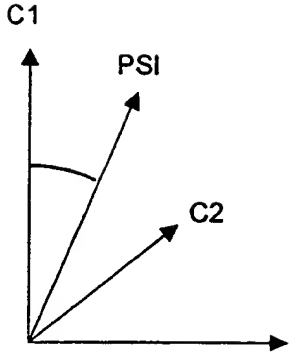


FIG. 10b

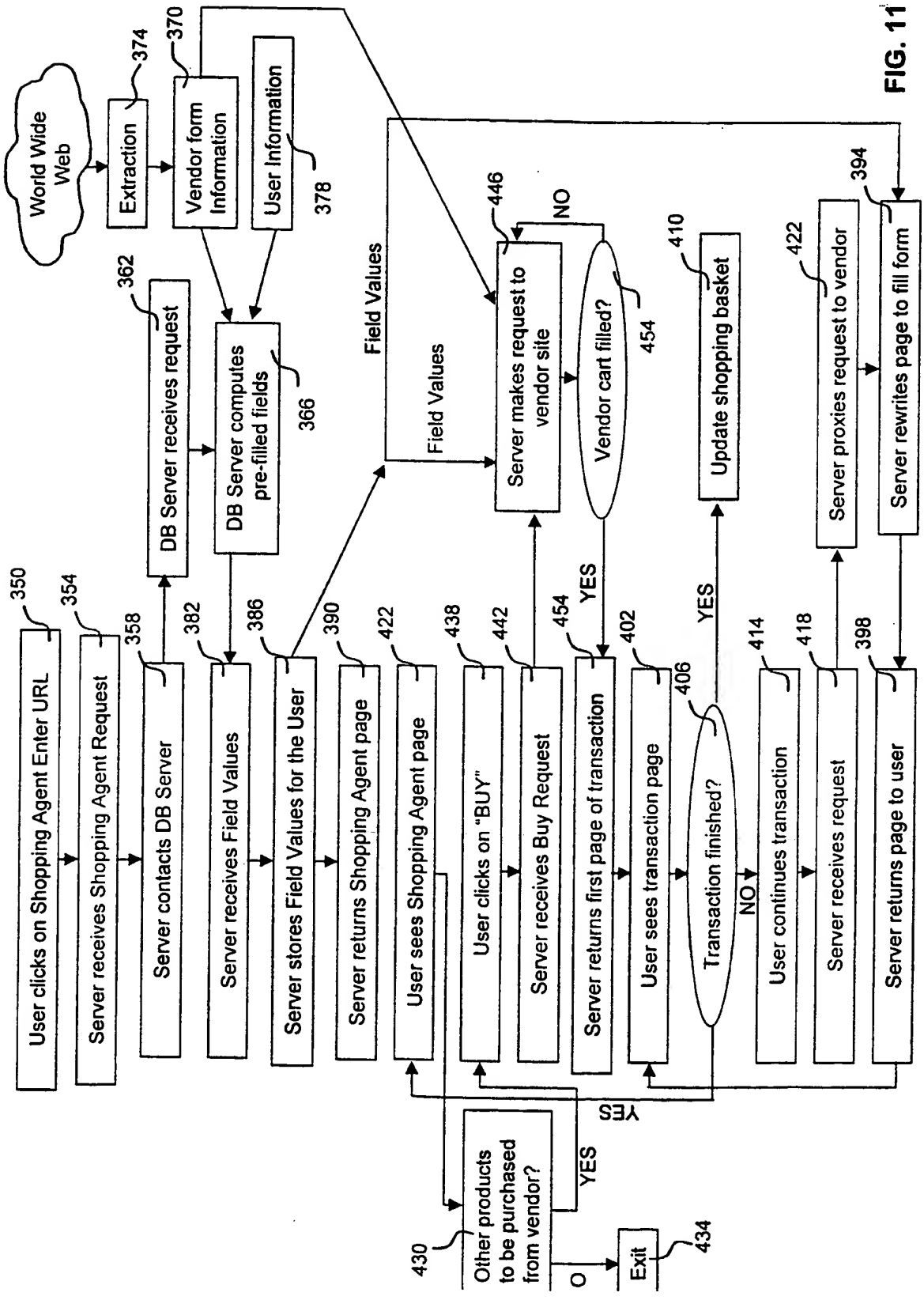

World Wide Web
Extraction
374
Vendor form Information
370
User Information
378
362
DB Server receives request
DB Server computes pre-filled fields
366
Field Values
Field Values
446
NO
Server makes request to vendor site
Vendor cart filled?
454
410
Update shopping basket
422
Server proxies request to vendor
394
Server rewrites page to fill form
350
User clicks on Shopping Agent Enter URL
354
Server receives Shopping Agent Request
358
Server contacts DB Server
382
Server receives Field Values
386
Server stores Field Values for the User
390
Server returns Shopping Agent page
422
User sees Shopping Agent page
438
User clicks on "BUY"
442
Server receives Buy Request
454
YES
Server returns first page of transaction
402
User sees transaction page
406
Transaction finished?
YES
NO
414
User continues transaction
418
Server receives request
398
Server returns page to user
YES
430
Other products to be purchased from vendor?
YES
NO
Exit
434

FIG. 11

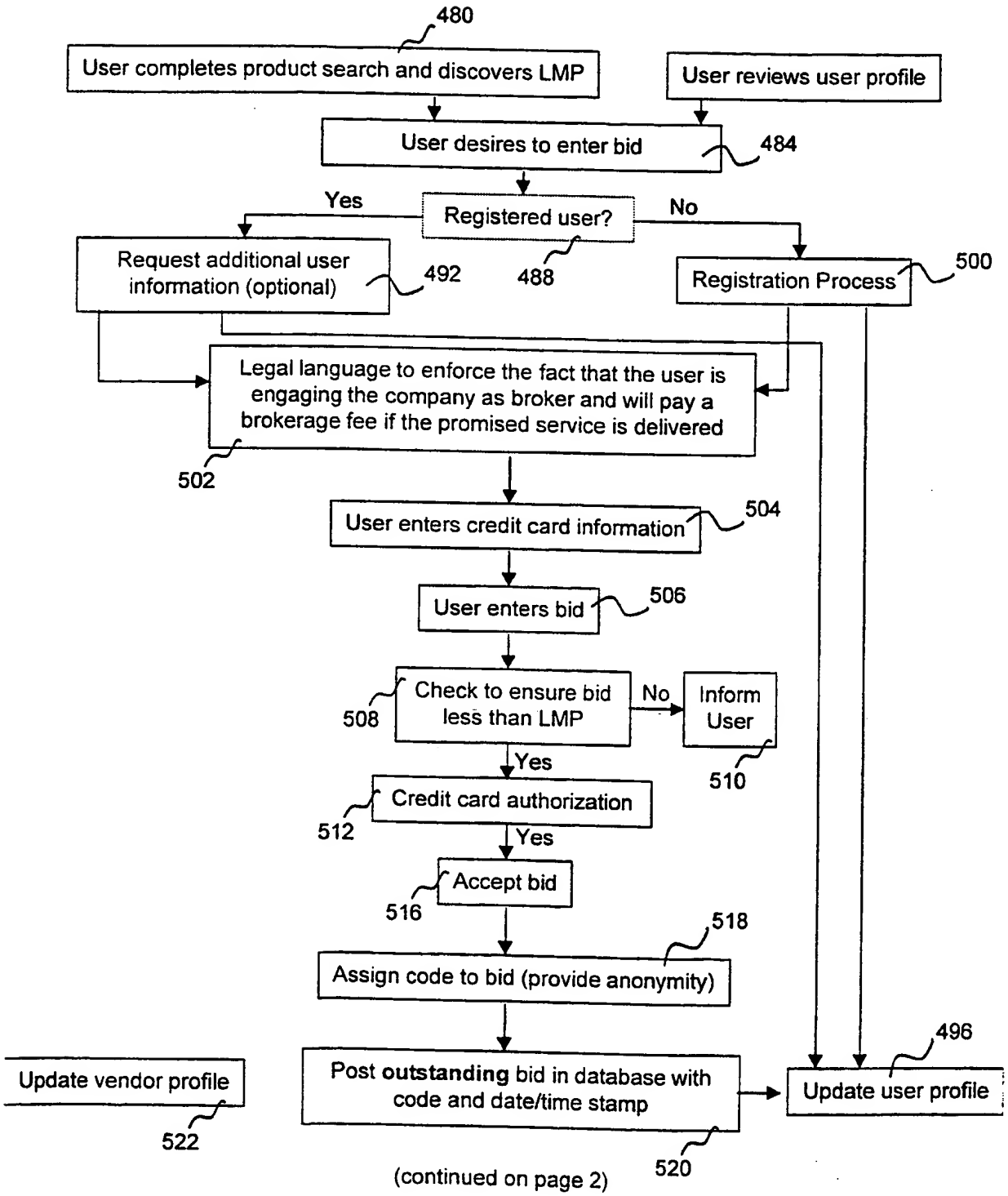
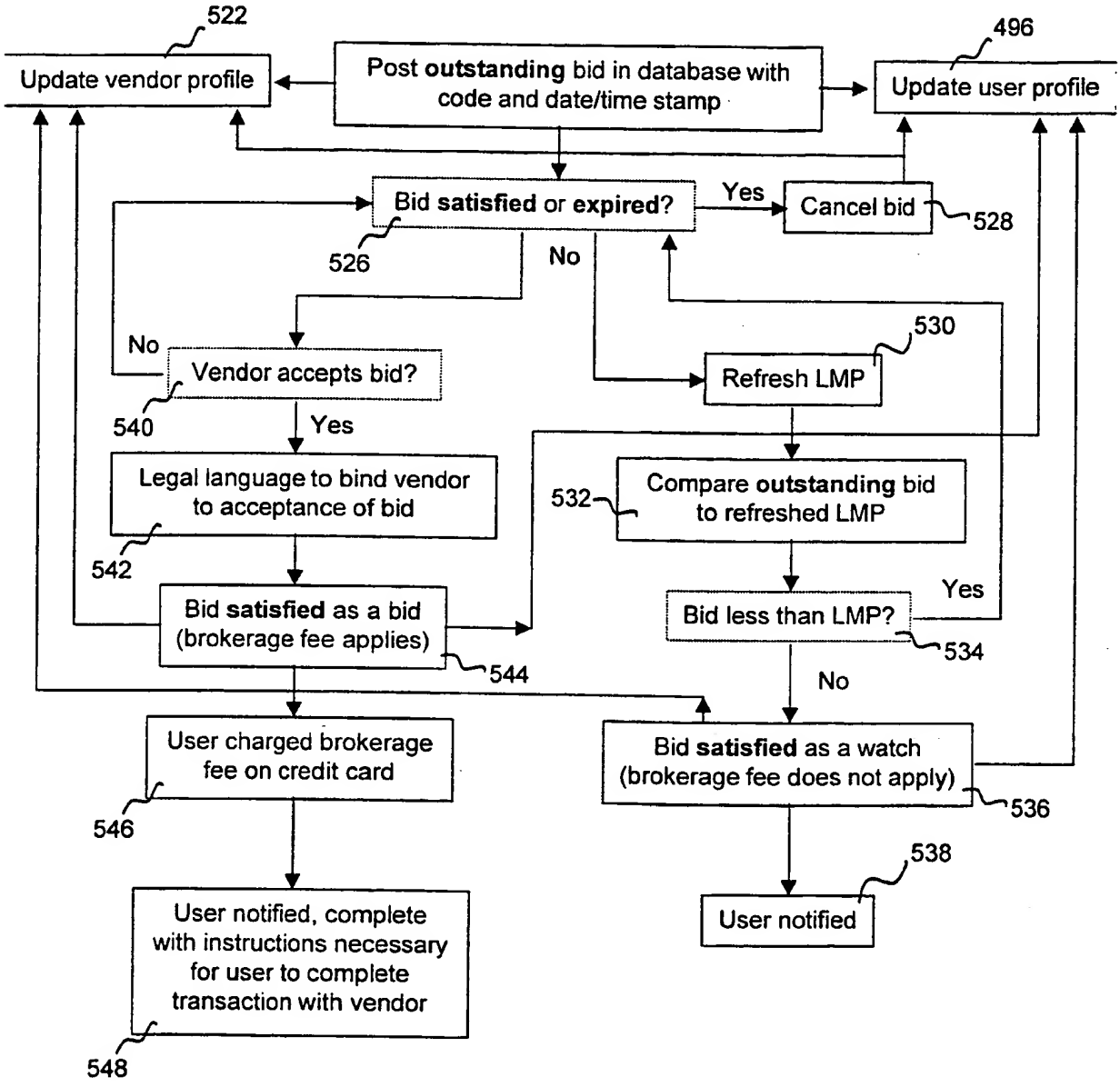

480
User completes product search and discovers LMP
User reviews user profile
User desires to enter bid
484
Yes
Registered user?
No
488
Request additional user information (optional)
492
Registration Process
500
Legal language to enforce the fact that the user is engaging the company as broker and will pay a brokerage fee if the promised service is delivered
502
User enters credit card information
504
User enters bid
506
508
Check to ensure bid less than LMP
No
Inform User
510
Yes
512
Credit card authorization
Yes
516
Accept bid
518
Assign code to bid (provide anonymity)
496
Update vendor profile
522
Post outstanding bid in database with code and date/time stamp
Update user profile
520
(continued on page 2)

FIG.12

(Continued from page 1)



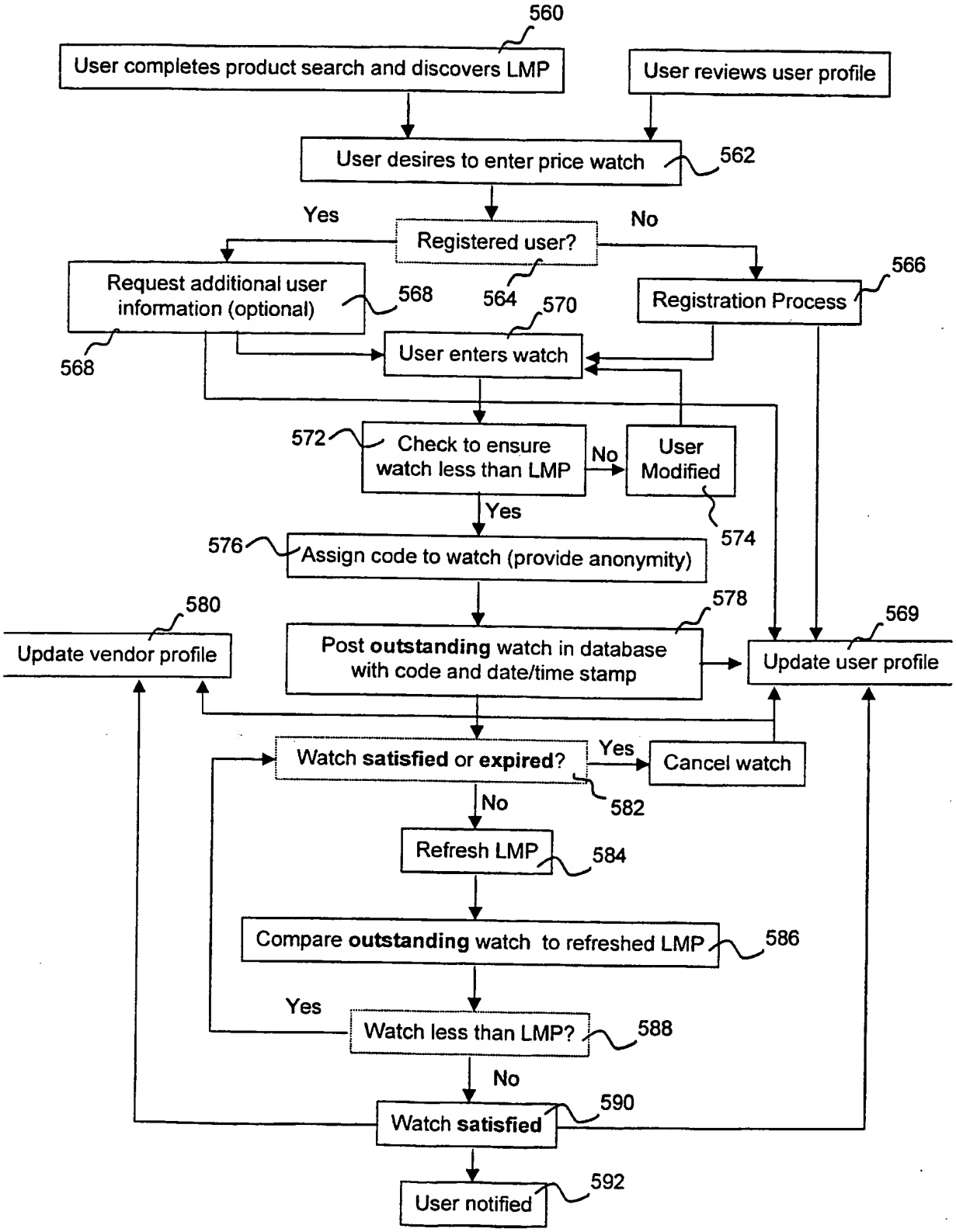

560
User completes product search and discovers LMP
User reviews user profile
User desires to enter price watch
562
Yes
Registered user?
No
564
Request additional user information (optional)
568
568
Registration Process
566
570
User enters watch
572
Check to ensure watch less than LMP
No
User Modified
574
Yes
576
Assign code to watch (provide anonymity)
580
Update vendor profile
578
Post outstanding watch in database with code and date/time stamp
569
Update user profile
Watch satisfied or expired?
Yes
Cancel watch
582
No
Refresh LMP
584
Compare outstanding watch to refreshed LMP
586
Yes
Watch less than LMP?
588
No
Watch satisfied
590
User notified
592

FIG. 13

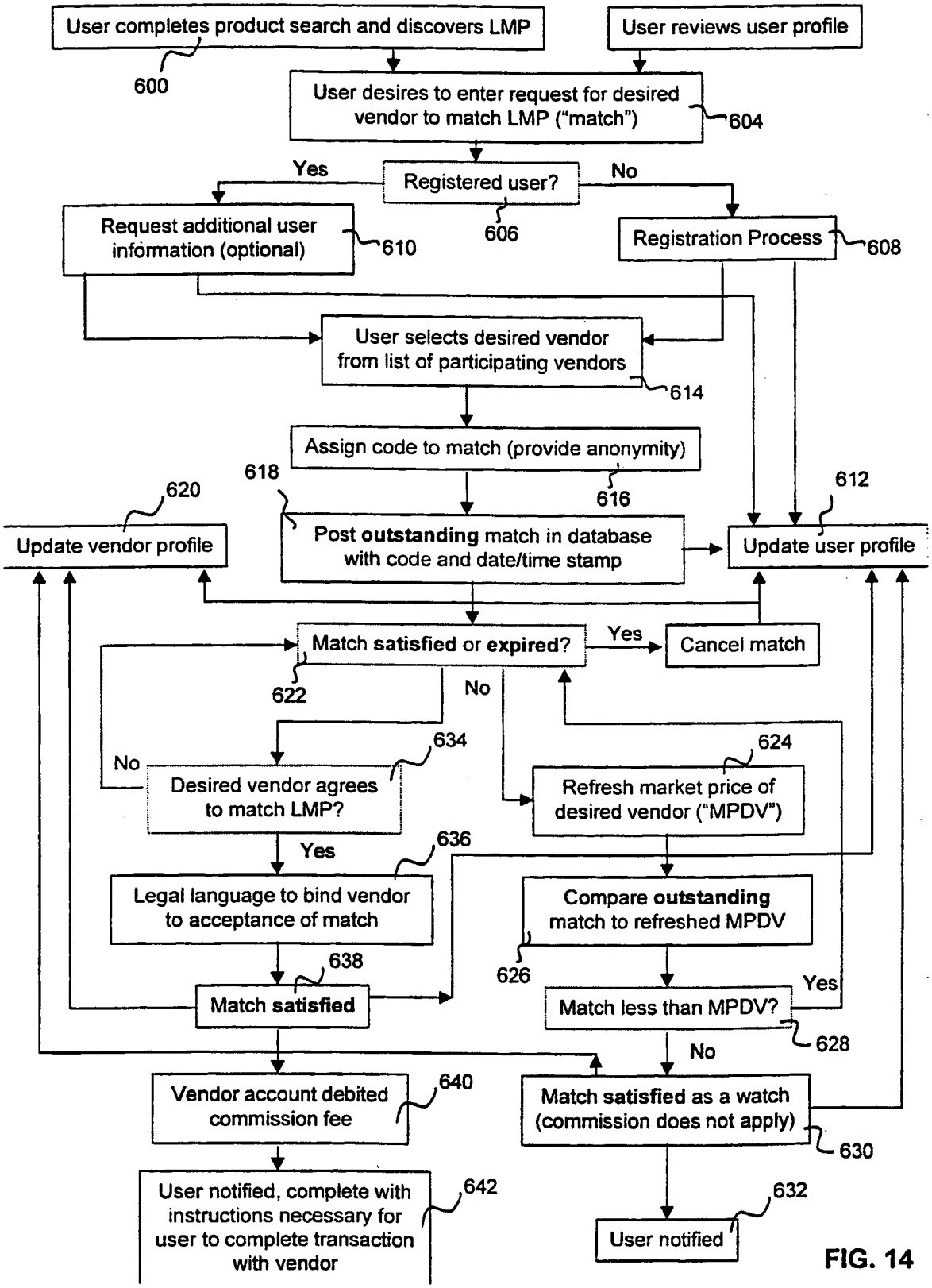

User completes product search and discovers LMP
600
User reviews user profile
User desires to enter request for desired vendor to match LMP ("match")
604
Registered user?
606
Yes
No
Request additional user information (optional)
610
Registration Process
608
User selects desired vendor from list of participating vendors
614
Assign code to match (provide anonymity)
616
618
620
612
Update vendor profile
Post outstanding match in database with code and date/time stamp
Update user profile
Match satisfied or expired?
622
Yes
Cancel match
No
634
No
Desired vendor agrees to match LMP?
624
Refresh market price of desired vendor ("MPDV")
Yes
636
Legal language to bind vendor to acceptance of match
Compare outstanding match to refreshed MPDV
626
638
Match satisfied
Yes
Match less than MPDV?
628
No
Vendor account debited commission fee
640
Match satisfied as a watch (commission does not apply)
630
User notified, complete with instructions necessary for user to complete transaction with vendor
642
User notified
632

FIG. 14